Universidade Estadual de Campinas

Instituto de Computação

Lucas Oliveira David

Connoisseur: Provenance Analysis in Paintings

Connoisseur: análise de procedência em pinturas

CAMPINAS

2019

Lucas Oliveira David

Connoisseur: Provenance Analysis in Paintings

Connoisseur: análise de procedência em pinturas

Dissertação apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Thesis presented to the Institute of Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Computer Science.

Supervisor/Orientador: Prof. Dr. Anderson de Rezende Rocha

Este exemplar corresponde à versão final da Dissertação defendida por Lucas Oliveira David e orientada pelo Prof. Dr. Anderson de Rezende Rocha.

CAMPINAS

2019

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

David, Lucas Oliveira, 1993-
D28c Connoisseur : provenance analysis in paintings / Lucas Oliveira David. – Campinas, SP : [s.n.], 2019.

Orientador: Anderson de Rezende Rocha.
Dissertação (mestrado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Aprendizado de máquina. 2. Certificados de procedência – Simulação por computador. 3. Reconhecimento de padrões. 4. Processamento de imagens – Técnicas digitais. I. Rocha, Anderson de Rezende, 1980-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Connoisseur : análise de procedência em pinturas
Palavras-chave em inglês:
Machine learning
Certificates of origin - Computer simulation
Pattern recognition
Image processing - Digital techniques
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Anderson de Rezende Rocha [Orientador]
Romain Giot
Paolo Bestagini
Data de defesa: 31-05-2019
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-8793-7300
- Currículo Lattes do autor: http://lattes.cnpq.br/8261642079102206

Universidade Estadual de Campinas

Instituto de Computação

Lucas Oliveira David

Connoisseur: Provenance Analysis in Paintings

Connoisseur: análise de procedência em pinturas

Banca Examinadora:

- Prof. Dr. Anderson de Rezende Rocha
  Universidade Estadual de Campinas
- Prof. Dr. Romain Giot
  Université de Bordeaux
- Prof. Dr. Paolo Bestagini
  Politecnico di Milano

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 31 de maio de 2019

Agradecimentos

Primeiramente, agradeço meus pais, Nara Teresinha de Oliveira David e Wilson de Oliveira David, aos professores e colegas com quem convivi durante minha vida, por comporem a base imprescindível na minha formação e desenvolvimento.

Agradeço ao professor Anderson de Rezende Rocha pelo apoio e orientação durante meus estudos e pesquisa; e aos meus colegas no RECOD pelo constante suporte, sem o qual este trabalho não seria possível.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

O crescimento de arte no meio digital tem, sem sombras de dúvida, democratizado o acesso ao conteúdo pelo público em geral. Entretanto, esse crescimento também implicou no indesejado aumento no número de falsificações e desinformação sobre conteúdos acerca de arte. Nesta linha, aprendizado de máquina pode ser utilizado para automatizar a organização e identificação de obras de arte em relação à sua procedência, auxiliando especialistas e usuários comuns na obtenção e validação de obras de arte. Empregamos neste trabalho estratégias baseadas em redes convolucionais para identificar e classificar artefatos digitais relacionados à arte. Primeiramente, pinturas de van Gogh são usadas para explorar e refinar estratégias capazes de discriminar seus padrões de pincelada. Múltiplos testes de conjuntos de dados cruzados são executados a fim de validar o método mais promissor encontrado. Os resultados indicam que atingimos uma drástica melhora em performance enquanto produzindo uma leve melhora em pontuação (90.99% acurácia em nível de segmento de pintura, 95.52% acurácia em nível de pinturas), quando comparado à estudos anteriores sobre o mesmo conjunto de dados. Estendendo nosso trabalho a partir da análise sobre van Gogh para um maior escopo, consideramos o conjunto de dados Painter by Numbers, onde expandimos nossa estratégia para o cenário multi-classe, onde buscamos distinguir pinturas divididas em 1.584 diferentes autores, 135 estilos e 42 gêneros. Propomos um método que combina informação dos três grupos de classes em um único discriminador de autoria, atingindo a ROC AUC competitiva de 0.91361 sem a aplicação de transformações potencialmente destrutivas sobre os padrões de pincelada que poderiam salientar características artificiais adjacentes, como brilho, contraste, escalas e objetos.

Abstract

Increasing digital art has without a doubt democratized the access to art content to the public at large. It has had, however, resulted in an inadvertent growing number of forgeries and misinformation around art content. In this vein, machine learning can be used to automatically organize and identify art content with respect to its provenance, aiding experts and regular users to retrieve and validate art pieces. In this work, we employ convolutional networks-based strategies to identify and classify art-related digital artifacts. Firstly, van Gogh paintings are used to explore and refine strategies capable of discriminating the brushstroke pattern of van Gogh. Multiple cross-dataset tests are performed in order to further validate the most promising method. We achieve significant performance improvements while slightly increasing accuracy (91% patch-level, 95.5% sample-level) when compared to previous studies in the same dataset. Extending our work from van Gogh analyses to a much broader setup, we consider the Painter by Numbers dataset, in which we expand our strategy to a multi-class scenario, seeking to distinguish paintings from 1,584 different authors, 135 art styles and 42 genres. We propose a method that combines information from these three class-groups into a single authorship discriminator, achieving the competitive ROC AUC of 0.91 without any transformations that could potentially damage the brushstroke patterns and emphasize adjacent features, such as brightness, contrast, scales and objects.

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area Under the Curve
BoW	Bag of Words
CNN	Convolutional Neural Network
CAE	Convolutional Autoencoder
FINE	Information Nonparametric Embedding
FC	Fully-connected Network
HMM	Hidden Markov Model
ISOMAP	Isometric Feature Mapping
LDA	Linear Discriminant Analysis
ML	Machine Learning
MDS	Multidimensional Scaling
MLP	Multilayer Perceptron
QMF	Quadrature Mirror Filters
PCA	Principal Component Analysis
ROC	Receiver Operating Characteristic Curve
SIFT	Scale-invariant feature transform
SPD	Steerable Filter Decomposition
SVM	Support Vector Machine

Contents

Chapter 1

Introduction

Art holds history. It tells interesting stories throughout the centuries to the ones that interact with it. As human beings, we seek to understand these stories in their completeness. This is not a trivial endeavor: with context changing, misinformation and degradation caused by the passage of time, important data necessary to the understanding of art works might become obscure, or even lost. To help us fill in the gaps, we resort to the idea of art connoisseurship.

Connoisseurship might be understood as an ability held by connoisseurs. In the context of art, a connoisseur is the individual who seeks to contribute to art scholarship by identifying works regarding their date, provenance and authorship [2]. This ability is usually acquired through extensive study over an artist's work, to which point recognizing it would be natural, or intuitive [3]. Of course, in practice, connoisseurs deliberate based not exclusively onto connoisseurship, but also on any evidence available.

In this vein, this work studies and attempts to improve the computational perception of art, specifically over the domain of paintings. We envision it potentially improving the decision-making process in a diverse set of related tasks, such as art style classification, automated dataset annotation and even facilitating painting connoisseurs to detect fake pieces and art fraud, an ever-growing problem nowadays, especially with the advent of digital art. This will be accomplished with machine-learning models optimized for the task at hand through training, which strongly resembles the very idea of connoisseurship: consider a mathematical model of multiple parameters and capable of yielding responses based on an input signal (e.g., image) and these same parameters, as well as a *loss* function capable of measuring the "incorrectness" level of the responses. Such model is submitted to what is known as "training", where it is presented with multiple work samples (paintings) and has its parameters adjusted so the *loss* is reduced to a minimum. In this process, the model analyzes samples considering their mathematical properties, extract features that concisely represent them and adjust itself in an attempt to either distinguish samples into groups that reflect authorship, genre or style similarity; or find a representation for the paintings set such that humanly recognizable patterns would emerge from it.

Over the last decade, Convolutional Networks (or simply CNNs) [4] have been successfully applied to feature extraction in images, resulting in the creation of models capable of impressive performance. However, much of the work done thus far have focused on photographs [5, 6], leaving artwork almost uncharted. When handling paintings, researchers

have applied their efforts to extract features with computer vision techniques and feeding those to conventional machine-learning models, such as Multilayer Perceptron Networks (MLP), Support Vector Machines (SVM), and k-Means. Although effective in many different setups, properly capturing specialists' knowledge and expertise regarding a painter's style and translating that knowledge into features is not straightforward.

Taking a different path in this research, we set forth the objective of designing and deploying an approach to automatically extract discriminative features from large painting collections and learn specific nuances and attributes of a given painter, in order to distinguish one's work from the others' and to automatically attribute paintings of unknown authorship to the known classes. This process is known in the forensic literature as authorship attribution and it has been applied before, for example, to literary texts [7] (matching a text to an author based on the style in which it is written), digitalized documents [8] (matching a digital document to a scanner) and more recently to printers [9] (matching a printed document to its source printer).

The remaining of this work is organized as follows. Chapter 2 discusses pertinent concepts and previous work that approached art considering one or more of the following topics: organization, visualization, authentication, authorship, and style analyses. Chapter 3 discusses the van Gogh authorship recognition problem, subdividing into the methodology 3.1, experiments 3.2 and discussion 3.3 sections. Chapter 4 repeats the same organization of Chapter 3, but will expand our approaches to a multi-class authorship matching problem, containing many distinct artists and styles. Finally, our conclusions are presented in Chapter 5.

Chapter 2

Related Concepts and Review of Literature

In this chapter, we first present the related background used to formulate the main aspects of this work. A brief review of the literature follows, aggregating the work performed thus far by its main objective or task: embedding 2.2, painting authentication 2.3, authorship analysis 2.4, style attribution 2.5 and style transfer 2.6. Within each section, the studies are sorted by the “approaching" used and its chronological order. Finally, we use Section 2.7 to discuss further explorations that may benefit art analysis in the machine-learning scope.

2.1 Artificial Neural Networks and Deep Learning

The developments of artificial intelligence (AI) and machine-learning (ML) are indeed astonishing. They are easily perceived when considering how intelligent agents changed from being able to play games using search algorithms [10] to integrating self-driving systems in automobiles [11]. From these developments, we draw a very important one: how machines perceive the world.

In the early days, AI was developed to tackle problems that are difficult to solve for human beings, but relatively easy to be described through a list of formal, mathematical rules [4]. The responsibility of modeling the problem was therefore entirely of the AI designer. Of course, such separation imposed limitations on what could be solved and the solution's effectiveness, as intelligent agents are bound to act according to this very limited perception.

Going in a different direction, modern machine-learning attempts to build its perception by extracting meaning directly from raw data. A way to achieve this is through Deep Convolutional Networks, which gained much momentum in the last decade given the positive empirical results it produced. Nowadays, deep learning is undoubtedly one of the most prominent approaches for extracting semantic information from raw data (e.g., images), leading to almost-human classification levels in some competitions [12].

Artificial neural networks (ANNs) are statistical models loosely based on biological neural networks, in particular the brain. Given their generic nature, ANNs can perform

many distinct tasks useful to machine-learning: transform, classify, regress etc. Deep learning refers to all machine-learning concepts that relate to deep neural networks, which in turn are nothing more than the stacking of multiple layers in an artificial neural network [4].

2.1.1 Fully-Connected Networks

A fully-connected network (FC) is a specific ANN in which each unit is connected to all (and only) units of the previous and following layers. Fig. 2.1 illustrates the idea with a 2-layer FC.

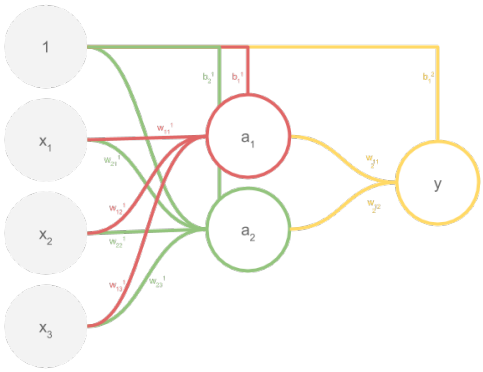


Figure 2.1: A fully-connected artificial network.

Let $\boldsymbol{x} \in \mathbb{R}^n$ be a sample in the dataset, $\boldsymbol{w}_i^j \in \mathbb{R}^n$ a vector of weights, $b_i^j \in \mathbb{R}$ a factor named “bias" and σ a function applied element-wise to tensors of arbitrary rank or size, commonly known as “activation". The output of a unit i in a given layer j can be written as:

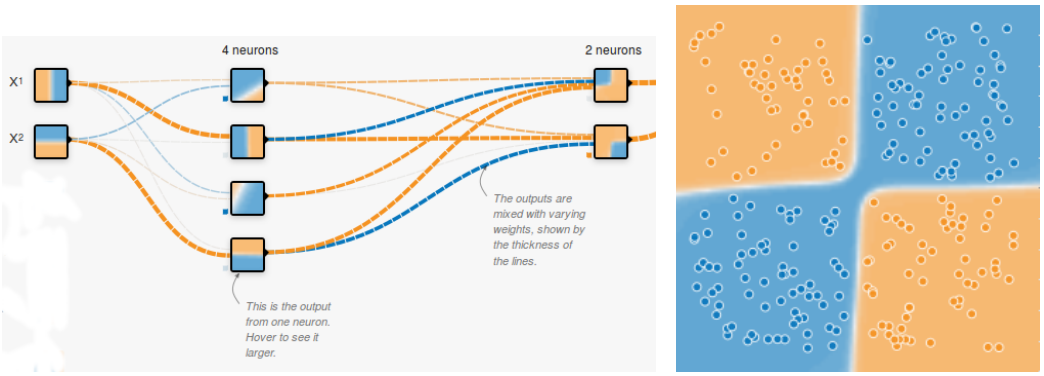
$$y_i^j = \begin{cases} \sigma(\boldsymbol{w}_i^j \cdot \boldsymbol{y}^{j-1} + b_i^j) & \text{if } j > 0 \\ \sigma(\boldsymbol{w}_i^j \cdot \boldsymbol{x} + b_i^j) & \text{if } j = 0 \end{cases}$$

More generally, if $\boldsymbol{W}^j = [\boldsymbol{w}_0^j, \boldsymbol{w}_1^j, ..., \boldsymbol{w}_n^j]^\top$ and $\boldsymbol{b}^j = [b_0^j, b_1^j, ..., b_n^j]^\top$ contain the parameters associated with all units in layer j, then layer j's output can be described as:

$$\boldsymbol{y}^j = \begin{cases} \sigma(\boldsymbol{W}^j \cdot \boldsymbol{y}^{j-1} + \boldsymbol{b}^j) & \text{if } j > 0 \\ \sigma(\boldsymbol{W}^j \cdot \boldsymbol{x} + \boldsymbol{b}^j) & \text{if } j = 0 \end{cases} \tag{2.1}$$

During supervised training (in which each sample $\boldsymbol{x}$ is associated with a target label $\boldsymbol{s}$), FC networks will adjust their set of trainable parameters θ (e.g. $\boldsymbol{W}$ and $\boldsymbol{b}$) to combine the output of the units in such a way to create decision boundaries, which can be used to discern samples or estimate (through regression) their values. As the output of the first layer represents the sample itself, one can interpret this whole process as an evaluation of the samples with respect to a predetermined set of characteristics. Fig. 2.2 illustrates a neural network with two hidden layers, containing four and two units, respectively, and the decision regions created by its configuration.

The parameters adjustment is commonly performed through a process known as *Gradient Descent* (or some variant of it), in which the variables are updated considering the opposite direction of the gradient of a predefined loss function $J(\theta, \boldsymbol{y}, \boldsymbol{s})$. Therefore, J must be differentiable and capable of measuring how distant are the predictions made by the model and the ground truth.



(a) A fully-connected neural network. (b) The decision boundaries.

Figure 2.2: Example of a fully-connected network and its decision boundaries used to classify samples between two distinct classes. Figures extrated from "A Neural Network Playground," *Tensorflow*. Available at: playground.tensorflow.org. License: Apache 2.0.

Although adequate to interpret low-dimensional or semantic data, fully-connected networks create difficulties when applied to raw data, such as image pixels in high dimensional spaces. Some of these can be briefly listed here: the difficulty in scaling, as high-quality images (usually represented by 3-rank tensors of high dimensionality) would require many parameters for every and each hidden unit; the disregard for spatial structures, as images are flattened and every pixel pair becomes equidistant; the strong sensitivity to translation, rotation and scaling, as the parameters are adjusted to fit a pattern of completely determined disposition.

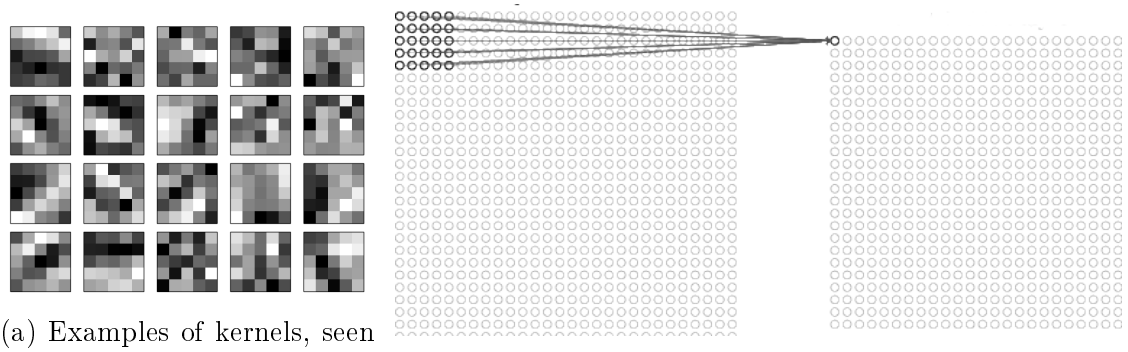
2.1.2 Convolutional Networks

Inspired by the organization of visual cortex of animals [4], convolutional networks attempt to abstract the input signal through hierarchical interpretative levels. CNNs stack multiple layers on top of each other in such a way units in the bottom layers will respond to simple patterns (e.g., lines and patches), whereas top layers will compose theses responses to match more complex patterns (e.g., objects and faces). This hierarchical pattern matching structure can be efficiently built by using the **convolution** operation.

CNNs share many similarities with FCs, including how training is performed. The difference is that it replaces the dot product operation performed in Eq. 2.1 by the convolution between an input signal (e.g., image) and kernels (or templates), as defined in Eq. 2.2.

$$(I * K)(i, j) = \sum_m \sum_n I(m, n) K(i - m, j - n) \tag{2.2}$$

Fig. 2.3a shows how these kernels look like, when interpreted as images; while Fig. 2.3b represents the sparseness of the operation: only the 5×5 up-left block contributed to the value of the first unit in the second layer.



(a) Examples of kernels, seen as gray-scale images.

(b) An input layer and a convolutional layer.

Figure 2.3: Examples of kernels and convolutional layers. Figures extracted from Michael A. Nielsen, "Neural Networks and Deep Learning," *Determination Press*, 2015. Available at: neuralnetworksanddeeplearning.com/chap6. License: CC BY-NC 3.0.

Notwithstanding, CNNs often intercalate **convolutional layers** with **pooling layers**. The pooling function (executed in pooling layers) down-samples the input signal $I \in \mathbb{R}^{n \times m \times c}$, reducing its dimensionality according to some operation (usually maximum or average), while eliminating some signal variance and, therefore, reinforcing translation invariance. Fig. 2.4 exemplifies this process.

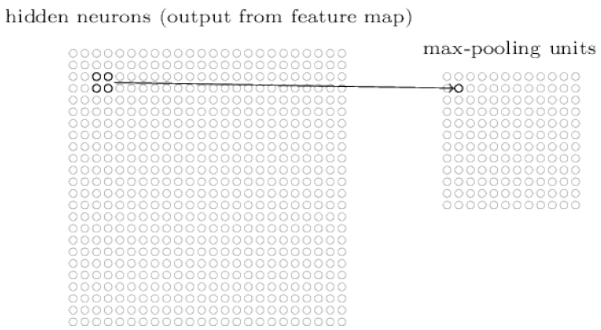


Figure 2.4: A pooling operation. Figure extracted from Michael A. Nielsen, "Neural Networks and Deep Learning," *Determination Press*, 2015. Available at: neuralnetworksanddeeplearning.com/chap6. License: CC BY-NC 3.0.

By using the operations defined above, three important properties manifest in convolutional networks: sparse interactions, parameter sharing and equivariant representations.

Sparse interactions refers to the fact that a unit in one layer is not completely connected to all units in the previous layer [4]. Fig. 2.3b illustrates this idea: each unit in the second layer is the output of the convolution between a signal (or image patch, if it is the first layer of the network) and a kernel (similar to the ones represented in Fig. 2.3a), both with same limited size.

Parameter sharing refers to the reapplication of the same parameters (kernel) to multiple patches of the signal [4]. The training phase, which adjusts the parameters, will focus on a very limited set, if compared to the dimensionality of the input signal. This simplifies the model and decreases training time necessary for convergence.

Equivariance to translation, which spontaneously follows parameter sharing. For this work, equivariance can be defined as the property of a function $f(\boldsymbol{x})$ to be affected in the exact same way of the input, when some change $g(\boldsymbol{x})$ is applied to it [4]. That is, $f(g(\boldsymbol{x})) = g(f(\boldsymbol{x}))$. This is specially interesting for images as translated patterns are processed similarly, making the recognition process insensitive to locality when used with *pooling*.

Fig. 2.5 illustrates the architecture of a neural neural network with one convolutional layer, one pooling layer and two fully-connected layers.

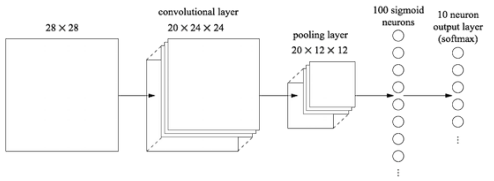


Figure 2.5: The diagram of a convolutional network architecture. Figure extracted from Michael A. Nielsen, "Neural Networks and Deep Learning," *Determination Press*, 2015. Available at: neuralnetworksanddeeplearning.com/chap6. License: CC BY-NC 3.0.

2.1.3 Contrastive Embedding

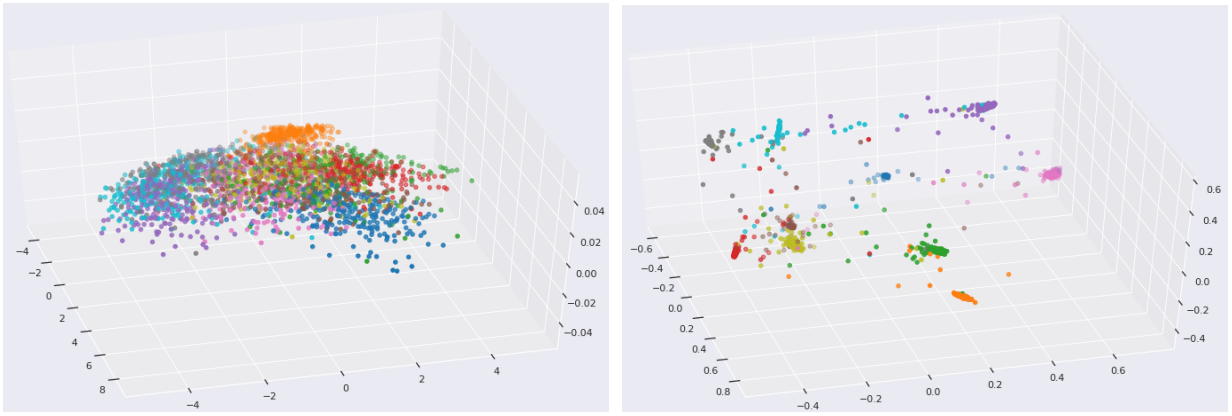
While many problems involve classifying and discriminating data samples, neural networks can also be optimized to embedding data. In such tasks, in which one is concerned with obtaining a meaningful representation from initially uninterpretable data, a well-known optimization process stands out: the contrastive embedding [13]. In it, a feed-forward embedding network is duplicated to accept pairs of samples from the original dataset. The two "legs" are joined with some neighborhood representative function and trained to minimize the distance between samples of the same class, while maximizing (limited to a margin commonly set to 1) the distance between samples of different classes.

$$\mathcal{L}(d, \boldsymbol{y}_1, \boldsymbol{y}_2) = \frac{(1-d)}{2}\|\boldsymbol{y}_1 - \boldsymbol{y}_2\|^2 + \frac{d}{2}\max(0, m - \|\boldsymbol{y}_1 - \boldsymbol{y}_2\|)^2 \tag{2.3}$$

Equation 2.3 illustrates the *contrastive loss* in its common form, where the l^2 distance function is employed to determinate dissimilarity between two feature vectors y_1 and y_2 that belong to the embedding space spawned by the network, when fed with two samples $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbf{X}$. Moreover, d represents the binary relationship between them: similar samples are represented by $y = 0$, while $y = 1$ describes a dissimilar pair [13].

When trained, the feed-forward model will be capable of projecting samples onto a more organized embedding space, with smoother transitions between samples, relative

to their classes. See Section 2.2 for a more detailed remarks on — art related — data embedding.



(a) The original feature space of the MNIST dataset (image pixels), reduced with PCA.

(b) The $\mathbb{R}^{128}$ embedded feature space of the MNIST dataset, reduced with PCA.

Figure 2.6: Example of a network trained over the MNIST dataset with the contrastive loss function. Colors indicate the ten different classes. Available at: gist.github.com/lucasdavid/15a35e53608875624c5ff8e5ab99f1ed.

2.1.4 Multi-objective Optimization

As artificial networks are trained, as briefly commented in Section 2.1.1, a single object function is optimized. However, in some cases, it is desirable to develop a model capable of inferring about multiple problems at once. For instance, when dealing with two adversarial problems, one might develop two separate models and obtain two different solutions that are locally optimal, for each problem, but incompatible when joined together. In such setup, evaluating the solution space considering both objectives can offer greater insight on the problem, resulting in better solutions in the larger scope. Another example is when performing multiple — but similar – tasks, in which some configurations (layers, units and weights) could be transported from one model to another. Here, one might reduce memory requirements, training and processing time by sharing parts of the model among all tasks.

It is difficult to determine an “optimal" solution in multi-objective scenarios, where solutions might conflict by presenting higher utility for some (but not all) objectives. Nevertheless, we begin by defining the set solution candidates. Let $J = \{J_1, J_2, \ldots, J_n\}$ be the set of all loss functions representing different objectives in a given problem and X the set of all possible solutions. A solution $\boldsymbol{x}^* \in X$ is said **Pareto optimal** if no other solution $\boldsymbol{x} \in X \setminus \{\boldsymbol{x}^*\}$ that improves $J_i, i \in \{1, 2, \ldots n\}$ can be adopted without increasing at least one function in $J \setminus \{J_i\}$. Furthermore, $\boldsymbol{x}^*$ is **weak Pareto optimum** if no other solution $\boldsymbol{x}$ would improve all functions in J [14].

As the addition of multiple differentiable functions will invariably produce a differentiable function, a simple way of handling multi-objective problems naturally follows: we define $T = \sum_i^n w_i J_i$. That is, the weighted sum of all individual objective loss functions.

Weights are assigned manually at the beginning of training, and will leak into the gradients and scale the modifications in the network's parameters suggested by each objective. This method will find Pareto optimal solutions as long as all weights are strictly positive [14], but presents an immediate limitation: understanding on the problem is required in order to set the weights to reasonable values, that directly match to the importance of each objective.

2.2 Embedding of Art Data

Dimensionality reduction methods are often employed when visualizing data or removing noise, as they transform the dataset by selecting or creating meaningful features (that strongly discriminate data) and discarding meaningless ones, which contribute little to data separation. This is essential for visualization, in specific, as human beings are limited to perceive only three dimensions and not all datasets can be embedded into a three dimensional space. For example, small paintings represented by 3-rank tensors with dimensions $224 \times 224 \times 3$ can be flattened into vectors contained in the $\mathbb{R}^{150,528}$ space. Projecting the first three values of each painting onto the $\mathbb{R}^3$ space would likely create a very poor representation, as light intensity of a single pixel is probably not a good factor to organize the data upon. Many embedding and data visualization methods find meaningfully visible representations by relying on the concept of sample dissimilarity according to meaningful metrics. They use it as a measurement to be preserved – at a local or global level – and hence preserving some structure from the original dimensional space while reducing its dimensionality. Among these methods, we remark Multidimensional Scaling (or MDS): a very well-known unsupervised, linear dimensionality reduction method that eliminates features while maximizing data variance. MDS has been used to embed art onto low-dimensional spaces in [15–17], as we describe next.

Image decomposition based on separable quadrature mirror filters (QMFs) [18] was performed in [15], resulting in a representation in which the frequency space was split into orientations and scales, while exhibiting statistical properties that can be exploited. A statistical model was built, composed by the mean, variance, skewness and kurtosis of the subband coefficients of the acquired representations; as well as a set of statistics that describe the higher-order correlations that exist within this image decomposition. Finally, Multidimensional Scaling (MDS) [19] was used to embed the 72-dimensional feature vectors into the $\mathbb{R}^3$. The authors validated their method on two domains: Bruegel's and Perugino's works. Using eight authentic Bruegel's paintings and five known imitations, the method generated a representation that clearly placed Bruegel's work into a cluster bounded by a sphere, and all imitations outside of this boundary. As for Perugino's, six patches containing a painted face, each was clipped from the "Madonna With Child" painting. From the representation found, the authors concluded that at least four different painters contributed to that art piece.

In turn, Bressan, Cifarelli and Perronnin [16] followed the Fisher kernel framework and extracted features using Scale-Invariant Feature Transform (SIFT) [20] and local color statistics over a dataset built from the harvest of paintings from Google and Yahoo!

image search, containing 51 different painters and at least 25 images per painter[1]. The features were later used to infer a metric describing the relationship strength between artist pairs, which in turn was used as a similarity measure to embed the artists onto a 2-dimensional space representation using MDS. The authors validated their metric by manually comparing it to the historical knowledge regarding the painters. They additionally observed a strong correlation of the embedding onto a 1-dimensional space to painters' birth date distribution. Unfortunately, it is not mentioned a significant name for the dataset or if it is available, making these results hardly reproducible.

Finally, Qi, Taeb and Hughes [17] worked over datasets containing van Gogh's and non-van Gogh's works (IP4AI1 and IP4AI2). Using the Fisher information distance [21] as a dissimilarity measure between brushwork pairs as input to MDS (a process which is also known as Fisher Information Nonparametric Embedding, or FINE), the authors were able to produce a low-dimensionality embedding which could be interpreted as a similarity map between paintings, possibly aiding art experts studying authorship and dating questions. The validation of the results was done by feeding the embedded data to a support vector machine classifier (SVM), which achieved 87.69% and 85% accuracy in IP4AI1 and IP4AI2, respectively.

The works presented above focused on organizing the information within the data by employing MDS over a set of features extracted from raw pixels or transformations of those, in a completely unsupervised pipeline. This choice is understandable, considering the lack of massive, annotated art datasets at the time. Since then, however, databases containing great amounts of metadata along with the paintings — such as WikiArt [22] — were made available. A high number of samples brings new challenges: a linear low-dimensional embedding, might collapse onto itself, creating a confusing representation if the data followed a non-linear distribution in the original space. On such cases, one possible solution is to assume that the data roughly lies on a non-linear manifold, and explore the entailed properties with non-linear dimensionality reduction methods (e.g., ISOMAP, Kernel PCA). Furthermore, metadata available can be incorporated in the embedding process (using, for example, Linear Discriminant Analysis, or simply LDA), creating multiple representations considering different characteristics of interest. Moreover these works' visual representations were constrained to only express similarities between a predefined set of features (e.g., brushwork, color histograms). A different path would be to use CNNs to automatically extract features, whereby clustering and visualization might express relationships between authors relative to unexpected, yet discriminative, characteristics.

2.3 Painting Authentication

Although visualization in itself already presents benefits to art connoisseurs, the authentication of a painting can also be included in the automated art analysis pipeline, creating agents able to distinguish – up to some level of accuracy – works of a specific artist.

Firstly, it is worth mentioning the work of three different groups presented in [23], in

[1] Unfortunately, no further indications were given regarding where to find such dataset.

which multiple decomposition methods and classifying models are used to extract brush stroke patterns from 101 paintings and discriminate them into van Gogh's and non-van Gogh's work.

The first group (Penn State) separated paintings into patches and extracted wavelet-based textural features using D4 orthonormal wavelet transform and geometric features of strokes (e.g., length, orientation, average curvature), computed from an edge-detection-based method. These features were later given to a Hidden Markov Model (HMM) [24], modeling a probabilistic process, which describes similarity between painting patches. A system was finally developed, allowing users to compute the "distance" from any painting P to any other P' in the dataset (or any subset of paintings) by averaging the Mallows distance from each patch in P to its closest patch in P'. Some inconsistencies were found when evaluating the descriptive model: two authentic van Gogh's were among the five test paintings farthest from a cluster composed by 23 authentic van Gogh's, leading the authors to conclude that the initial training data (the 23 paintings) might have not been sufficiently representative or that the comparison method could yet be improved.

The second group (Princeton) also utilizes HMM to model paintings into 108-dimensional feature vectors, but defines the similarity between paintings as the sum of weighted $l2$-distances between each pair of patches of these same two paintings, in which each feature weight is proportional to that features' contribution in discriminating van Gogh's and non-van Gogh's works. Finally, the group exploits MDS to find a representation for the paintings. A 3-dimensional one, in specific, adequately presented non-van Gogh's farthest from C_{vG}, the centroid of van Gogh's paintings. Using a classifier based on a simple threshold value for centroid-to-sample distance over the embedding found, the authors reached 84.2% in accuracy.

The last group (Maastricht) extracted the necessary features by employing Gabor wavelet filters and histogramming the resulting coefficients in multidimensional bins (6×4), forming 24-dimensional feature vectors. The choice for Gabor filters is due to their capability of capturing brushstroke patterns at multiple scales and orientations. Automatic classification was then performed over the feature space created, yielding most encouraging results: four out of the six non-van Gogh paintings were detected, at the cost of wrongly classifying two van Gogh paintings.

Shortly after, Irfan and Stork [25] attempted to authenticate Jackson Pollock's drip paintings (12, at total), achieving higher accuracy (81%) when adding fractal-based features to a previous set solely based on metrics such as genus and orientation energy. The authors concluded that fractal-based features were indeed adequate to describe Pollock's work. Moreover such features could still present some improvement in classification accuracy when combined with other features based on brushwork analysis.

An important idea supported by art connoisseurs was presented in [26]: highly detailed regions in paintings are compositions of small and calculated corrections, rather than a fluid brushwork spontaneously created by an artist, which contains their signature. In an attempt to consider only spontaneous brush strokes, patches were extracted from a painting and discriminated as foreground and background. Only the latter was considered during the feature extraction phase, similar to [17]. Authentication was performed through what the authors called the "outlier classifier", a model that simply compares if the

“pertinence level" of a sample to the van Gogh cluster is greater than a threshold value optimally set through training. Experiments showed substantial accuracy increase by employing this idea: when considering all patches, the outlier classifier achieved 80% of accuracy on IP4AI1 dataset and 75% accuracy on IP4AI2; whereas 92.31% and 91.25% of accuracy were achieved when considering only background patches of the IP4AI1 and IP4AI2 datasets, respectively.

A different authentication method for van Gogh's paintings is proposed in [27], based on some simple statistics of the geometric tight frame coefficients, boosting procedure for feature selection and an outlier classifier similar to [26], except that it now checks if the distance of the sample to the van Gogh's cluster center is bellow a threshold value to attribute that sample to van Gogh. Their method yielded 88.61% accuracy by selecting only the five most frequently observed features on the samples, which possibly implicated in noise removal. Although the work was limited to select features by their frequency, one can also efficiently eliminate noise through PCA or autoencoders [4], which have recently presented promising results in literature [28].

The authentication of Amadeo de Souza-Cardoso's work was studied in [29] through the analysis of not only brushwork with Gabor filters [30] and SIFT, but also of the pigments used in the art piece's composition with hyperspectral imaging combined with X-ray fluorescence analysis. From their experiments, the authors concluded that an analysis of pigmentation is of most relevance when identifying forgeries.

Finally, a pre-trained CNN (VGG-19) was used in [31] to transform equally-sized patches of paintings and a linear SVM to detect traits of van Gogh's work on those transformations. The decision whether a painting was a van Gogh's or not was taken through a voting system, placed at the end of the classification pipeline. Multiple approaches to combine the answers for the image patches were tested, but the most successful was “far": a painting was discriminated based on the classifier's most confident response among all patches (the farthest answer from the decision hyperplane). This result is specially intriguing: it resembles [26] as it also assumes some patches to be more descriptive than others, but differs by leaving the selection itself to the learner. The authors were able to predict with 93% accuracy whether or not a given sample was a van Gogh's work. In spite of the encouraging score, the work was restricted to binary discrimination and it is still uncertain how that would behave for multi-class attribution, in which a more realistic scenario considering multiple painters is analyzed. The work also fell short in properly training a network over a painting dataset containing van Gogh's work, which could possibly better fit the model to this problem and further increase accuracy.

Although fairly successful at their tasks, the previously mentioned works were limited to their painters' style characteristics and could present difficulties if extended to other painters. Yet, many interesting ideas presented can be leveraged and extended. We aim at exploiting in this work: the extraction of highly-descriptive areas in [26], the feature pre-selection/noise removal phase in [27], and the transfer learning employed in [31]. Of course, it is also desirable to not only increase accuracy in very specific learners, but also to expand the problem into a more generic (and difficult) formulation: authorship attribution.

2.4 Authorship Attribution

While authentication is focused on answering the question “is this painting from this specific author?", authorship attribution attempts to assign each painting (sample) to an author (label) from a set of possibilities. This problem is approached by many authors [32–37], as we detail next.

A dataset collected from various internet sources, composed by 513 paintings of nine different painters representing impressionism, abstract expressionism, and surrealism, was studied in [32]. The authors began by separating each painting into 16 equal-sized tiles, and proceeded to extract features from each of the patches and several transformations of them using a set of known algorithms (e.g., Radon transform features, Chebyshev Statistics, Gabor Filters), resulting in $16 \times 3,658$ image descriptors for each painting. The most promising results (77% in classification accuracy over all painters) were obtained with the “two-stage image classifier" procedure suggested by the authors, which would first classify a painting into a school of art and only then perform intra-school authorship attribution. Although an interesting idea, we have not seen other works exploiting this path further.

Meanwhile, Cetinic and Grgic [34] conducted experiments comparing authorship classification accuracy of models fed with three major groups of features – image intensity statistics, color-based features and texture-based features – over a dataset consisting of heterogeneous paintings, acquired from different websites and sources, varying in quality, density and size. The authors found out that texture-based features consistently presented the highest accuracy, although the combination of different approaches would still significantly improve the classification score.

Tamura textural features were used in [33], as they correspond to human visual perception, being useful for feature selection and texture analysis design. Among the six basic textural features available, only three were adopted: coarseness, contrast and directionality. Over a dataset of 100 Chinese paintings belonging to four artists, the authors observed a considerable classification accuracy increase by training an SVM fed with Tamura features (80%), when compared to the results obtained through classifiers trained over SIFT features (65%).

In order to study authorship attribution, 120 traditional Chinese paintings from six different painters were collected in [35]. An interesting classification pipeline was adopted, in which the most representative brush strokes are first extracted from a painting with edge detection methods and then sent to a simple CNN, resulting in a feature vector containing 3,000 features. Using a Sparse Group Lasso Classifier, the authors achieved 94% binary classification accuracy and 81% when considering all six painters. Although the brush pre-selection seems an interesting strategy to remove noise, it is not clear if the positive results were caused by this step or the small number of classes. It would be interesting to test this idea in a more challenging dataset.

Inspired by painting authorship, Thomas and Kovashka [36] used multiple CNN architectures and linear SVMs to perform authorship attribution on 181,948 photographs, belonging to 41 well-known photographers. The dataset was collected from multiple inter-

net sources and was made available by the authors[2]. Experiments suggested a pre-trained Hybrid-CNN – trained over both Places and ImageNET datasets [38], hence the name – achieved the highest accuracy (74%), outperforming the score obtained by low-level features extracted from methods such as GIST [39]. Surprisingly, the authors also reported the Hybrid-CNN achieving higher scores than PhotographerNET, a neural network of same architecture that was trained from scratch, using the photograph dataset at hand.

A CNN (PigeoNET, based on AlexNET [5]) was used to transform and classify samples from the Rijksmuseum Challenge [40] dataset in [37]. The dataset at hand was composed by 112,039 samples of digital photographic reproductions of artwork from 6,629 artists, including paintings, prints, drawings, sculptures and other artworks. The authors achieved promising results in authorship attribution: 60% of accuracy when considering 10 prints for each one of 673 painters; and 78.8% when considering 256 prints for each one of 29 painters. Additionally, visualization techniques were explored to determine regions of interest used when classifying a painting and to determine which regions characterize an author, further evaluating cases of misclassification. However, the work lacked tests regarding more recently developed network architectures, classifiers other than **softmax** (while still using the CNN as a transformer) and ignored many of the ideas presented in previous art (e.g., background patch pre-selection and noise removal). Furthermore, the behavior of this solution over other datasets that include only paintings is yet to be verified.

Viswanathan [41] used a modern convolutional network architecture (ResNet-18) to extract features and classify between the 57 of most represented artists in the Painters by Numbers dataset. The author achieved a 77.7% Top-1 and 97.3% Top-3 test accuracy, when the test set comprised 10% of the original Painter by Numbers dataset. Chen and Deng [42] tackled the same dataset, but narrowed it even further to only 15 artists (authoring at least 450 paintings each). Besides testing modern deep-learning techniques, the authors compared them to more classical algorithms (e.g., support vector machines and logistic regression) fed by GIST descriptors, Hu moments and color histograms. They have concluded that convolutional networks outperformed GIST+Hu Moments+SVMs by 6.6% Top-1 test accuracy, while requiring considerable shorter inference time. Although achieving enticing results, the authors have drastically reduced the difficulties and challenges from this set by considering a small fraction of the original attribution set, where 1584 artists were present.

2.5 Style Attribution

Similar to authorship attribution, the style attribution problem concerns in assigning a style to paintings, from a set of possible options. Onto this problem, we observe works such as [43–48].

Over a dataset of 353 paintings belonging to five artistic styles, Zujovic et al. [43] modeled paintings as feature vectors composed by Steerable Filter Decomposition (SPD), which is shown to exhibit properties of low-level processing in human eyes and is also

[2] people.cs.pitt.edu/~chris/photographer

translation-and-rotation-invariant; edge features, as edges might be highly informative regarding painting styles; and simple color histograms. Classification was performed over this representation, wherein each painting was assigned to one of the five genres considered using an AdaBoost classifier, achieving 68.3% accuracy.

Meanwhile, Kovashka and Lease [44] compared stylistic classification accuracy between humans and machines over a dataset collected from Wikimedia Commons, composed by 240 paintings of six authors. The "machines" hereby mentioned are models fed by multiple types of features. Among these, features extracted through the convolution of the painting and filters (some of which resemble strokes) from a bank achieved the highest agreement with humans, indicating that as a promising feature extraction technique. On the other hand, experiments by Bajcsy and Moslemi [45] also indicated color histograms in patches containing faces extracted from paintings as highly discriminative features.

With 490 paintings collected from the Mark Hadern's Artchive, Arora [46] composed a painting dataset equally distributed between seven categories. Classification was done with SVMs trained and tested with 5-fold cross-validation over multiple sets of features (Discriminative and Generative Bag of Words with color SIFT and opponent SIFT; as well as Classeme features). From the experiments, Classeme descriptors — a composition of many predefined basic classifiers previously trained, capable of identifying semantic-level information within the paintings — consistently presented higher accuracy score (65.4%) than the others, such as Discriminative BoW OSIFT, which achieved the second place, scoring 56.7%. The conclusion was immediate: Classeme descriptors were able to generate a meaningful representation, capable of achieving high classification scores at genre attribution and even outperform more traditional methods.

While only a very strict set of classes were considered by the authors above, the classification of 27 different art styles was approached by Bar, Levy and Wolf [47] over the Wikiart dataset, currently containing approximately 150,000 artworks created by 2,500 artists. The authors extracted features from classic methods (e.g., SPD, Color histogram, GIST), as well as PiCoDes (a very compact and efficient image descriptor for object recognition [49]) and a CNN (Decaf implementation of AlexNET). The experiments showed that the features extracted from the CNN could distinguish styles with higher accuracy (37%) than low-level ones (23%), even when the dimensionality of the vector space containing the CNN's output was reduced from 9216 to 405 features. Furthermore, combining these with PiCoDes resulted in a 6% increase in accuracy, yielding state-of-the-art results (43%) in 2014.

Finally, in 2015, Saleh and Elgammal [48] built a "unified" authorship, art style and genre classification framework. Once again over the WikiArt dataset, features were first extracted from multiple methods (e.g., GIST descriptors, Classeme features, PiCoDes and a CNN pre-trained over ImageNet dataset [5]). For each feature space extracted, a similarity metric specific for the task at hand (e.g., author, style or genre classification) was learned, where that metric induces a projector to a feature space optimized for its task. Having a metric learned for each feature type, vectors in their corresponding feature spaces were projected onto new optimized spaces. These new representations were then concatenated and fed to common classifiers. The authors achieved, up to our knowledge, the current state-of-the-art of 45.97% accuracy on the style attribution problem. Notwith-

standing, the authors used a rather simple architecture for the CNN, besides having its parameters set through transfer learning without performing any actual training or fine-tuning over Wikiart itself. In these conditions, the CNN could be generating unoptimized features, possibly insufficient to generalize the patterns of paintings style in Wikiart. Further improvements could therefore still be achieved by exploiting other architectures and proper training.

2.6 Style Transfer

In 2015, Gatys et al. posed the style transfer task as an optimization process of combined loss functions from a neural network [50]. The algorithm consisted of using a set of layers $L = \{l_0, l_1, \ldots, l_n\}$ of a convolutional network (previously trained for object recognition) to extract multiple features from a content image $\boldsymbol{C}$ and a style image $\boldsymbol{S}$. A third, varying image $\boldsymbol{O}$, would then be randomly initialized, and its content and style representations would be matched to the ones of $\boldsymbol{C}$ and $\boldsymbol{S}$, respectively.

For a given layer l_c, the content reconstruction loss is defined as:

$$\mathcal{L}_{\text{content}}(\boldsymbol{C}, \boldsymbol{O}, l_c) = \frac{1}{2} \sum_{i,j} (l_c(\boldsymbol{O})_{i,j} - l_c(\boldsymbol{C})_{i,j})^2 \tag{2.4}$$

Towards style representation, the authors start by defining the style content of a given layer l_s as the correlation between different filter responses l_s^1, l_s^2, ... l_s^f, extracted by Gram matrix applied to the vectorization of l_s's output:

$$\begin{aligned} G_{i,j}^{l_s}(X) &= \sum_k vec(l_s^i(X))_k vec(l_s^j(X))_k \\ E_{l_s} &= \frac{1}{4f^2M^2} \sum_{i,j} (G^{l_s}(\boldsymbol{O}) - G^{l_s}(\boldsymbol{S}))^2 \end{aligned} \tag{2.5}$$

Each term E_{l_s} is normalized by the size of input $l_s(X)$ and output feature spaces G^{l_s}, preventing earlier layers (with large input signals) and later ones (with higher filters) to output unbalanced values and, therefore, having an unmeasured impact on the optimization process. The style representation loss is finally defined as the weighted combination of individual losses:

$$\mathcal{L}_{\text{style}}(\boldsymbol{S}, \boldsymbol{O}, L) = \sum_{l \in L} w_l E_l \tag{2.6}$$

Content and style losses can now be combined into a single one $\mathcal{L}_{\text{total}}$ with simple or weighted addition. An optimization process (such as Gradient Descent) is used to tweak the pixels in image O in order to minimize $\mathcal{L}_{\text{total}}$, as presented in Section 2.1.4.

Since then, further exploration has been performed towards style transfer. In [51], the authors trained a network capable of reconstructing multiple input images into new ones, minimizing their style difference towards a style reference while maintaining their original content. This allowed faster style transfer, even though constrained to a specific

style. Wang et al. combined the Gram matrix texture representation with Local Pyramid Features to better reconstruct the subtleties in local patches and improve the quality of the generated image [52]. These works serve as evidence that the representation adopted by Gatsy et al. are indeed capable of capturing the underlying style structure in the painting. One might wonder if such patterns are sufficient to also indicate authorship and, furthermore, be used to discriminate paintings based on their creators.

Figure 2.7 exemplifies the transfer of multiple styles (first line) into content images (first column).

Figure 2.7: Example of style transfer applied to common photographs. The first row contains the three style reference paintings, from left to right: “Water Lilies", by Claude Monet, 1919; “Rocks with Oak Tree", by van Gogh, 1888 and “Stone bench in the garden of the hospital of Saint-Paul", by van Gogh, 1889. The content photos, contained in the first column, were either cordially ceded by the authors or freely distributed[3]. Style transfer was either performed using the code in neural-style-transfer repository[4] or with the deepart.io tool[5].

[3]Max Pixel. Old Architecture Arches. Available at: maxpixel.net/Old-Architecture-Arches-Round-Arch-Building-Arch-3416038

[4]GitHub. (2018) Neural Style Transfer. Available at: github.com/titu1994/Neural-Style-Transfer

[5]Style transfer tool based on the VGG19 network designed by Karen Simonyan and Andrew Zisserman. Available at: deepart.io.

2.7 Further Investigation

We highlight the importance of promoting a study over multiple domains and problems in art, while exploiting the computational resources currently available. Such study has potential to increase the classification accuracy of the solutions to practical problems described by modern datasets, which is still very limited; as well as help connoisseurs and art historians to better understand hidden relations built within painters, styles, genres and art in general, further validating the importance of computer science applied to art.

From the previously presented findings, it is clear that analyzing art with respect to more significant and complex properties rather than simple color or pixel-shifting statistical measurements is fundamental to further understand the mathematical models and characteristics intrinsically imprinted onto it. The prominent option of convolutional networks presents itself as an interesting option, which can be exploited through the study over multiple painting datasets, novel neural network architectures, learning methods and end-point estimators, while accounting for the interesting ideas presented thus far, as transfer learning from larger datasets, “multiple-stage classifiers", noise removal through dimensionality reduction, pre-selection of areas of interest and injection of artist-specific knowledge into our models. All of these aspects are subject of interest of our research presented in Chapter 3 and 4.

Chapter 3

Authorship Attribution in van Gogh Paintings

While the different approaches presented in Chapter 2 relate to divergent data sources and art groups, it is noticeable how van Gogh's has been sistematically analyzed, creating an enriched environment in which a wide range of methods can be more accurately compared. As such, we opted to start our studies with this painter as well.

In this work, exploratory research using multiple modern models (e.g., convolutional networks, contrastive siamese networks) was performed over the van Gogh 2016 dataset [31]. This dataset comprises 264 train and 67 non-balanced test high-quality captures of paintings, discriminated by the not van Gogh (nvg) and van Gogh (vg) labels. Leveraging the main classification strategy employed by Folego et al. [31] and insights from Qi and Hughes [26], we were able to achieve drastic performance improvements (compared with the former) while slightly increasing accuracy over the van Gogh 2016 dataset. Additionally, we push toward cross-dataset testing in order to further validate the findings and better estimate the generalization capacity of our method on a highly varying environment.

3.1 Proposed Methodology

In this section, we present details of the proposed methodology for exploring convolutional network-related techniques targeting authorship attribution in paintings.

3.1.1 Sampling Few Patches to Represent a Painting

While the ensemble of patches predictions as a fortification strategy yielded good results [31], it was also very resource demanding, as it required the analysis of almost an entire painting for decision making. This could prove itself even more troublesome for larger datasets.

The solution in [31] was re-implemented in order to retrieve the test patch-level accuracy (in opposite to the 94.03% painting-level accuracy reported in their paper): $54,092$ patches were extracted (approximately 205 patches per painting) and fed to the VGG19 network. Features extracted from the second fully-connected layer were then used to grid

search the best parameters and train a linear SVM model. The entire procedure took approximately three hours. The value of 82.40% found was an indicative that much of the method's accuracy was given by the accuracy upon deciding on patches alone, and not entirely dependent on the patch ensemble performed. This suggested that fewer patches could be extracted to represent each painting while maintaining the overall accuracy, and raised questions such as which sections of a determined painting to extract and how many patches would sufficiently represent the paintings provenance.

To show that deciding authorship based on a sub-sample of patches – in opposite of deciding upon all of them – does not severely affect accuracy, the method employed in [31] is re-applied onto van Gogh 2016 dataset several times, considering a different amount of patches each time. As the experiment proceeds, the model is trained and evaluated with a subset containing fewer patches to represent each painting.

Three different patch extraction methods were tested, and are described below:

3.1.1.1 Random

k points are randomly chosen in the painting and k non-disjointed patches centered on those points are extracted. It is expected from this method to comprise the most "unbiased" selection, resulting in a patch sample set with foreground/background, highly/low-detailed rate proportional to the original painting. It is also the cheapest (computationally-wise) selection strategy, as no deeper analysis on the input painting is necessary other than extracting its height and width. Figures 3.1 and 3.2 illustrate examples of this strategy.

Figure 3.1: Samples of patches extracted with the random strategy.

(a) “White House at Night", by Vincent van Gogh, 1890.

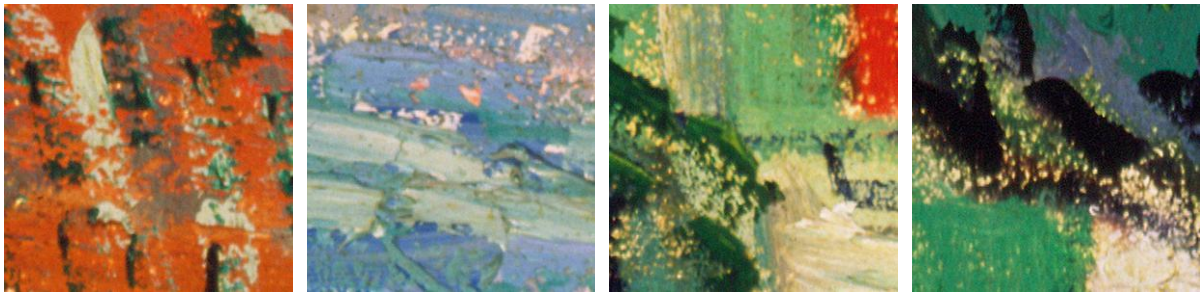
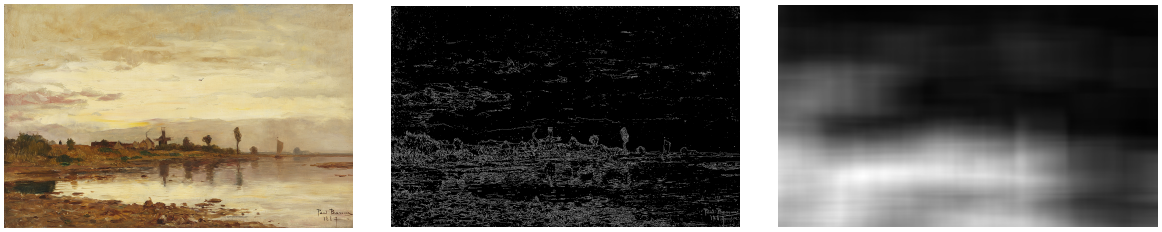


Figure 3.2: Examples of patches extracted from “White House at Night", using the *random* selection strategy.

3.1.1.2 Max-grad

In an attempt to select highly detailed sections, comprising a higher amount of shorter brushstrokes, in which the author has presumably spent most of their time, we define a selection strategy that is drawn to highly variant sections of the original painting.

Let $h, w \in \mathbb{N}$ be the desired patch height and width, respectively, and $s \in \mathbb{N}$ a stride factor. The Canny edge detection filter is used to generate an edge map from a grey-scale image of the painting (Figure 3.3b). Average pooling 2D with stride s is applied over this map in order to decrease memory and time requirements, and the result convolved with a $\mathbf{1}$ kernel of shape $h/s \times w/s \times 1 \times 1$. Finally, the **softmax** function is applied, which effectively blurs the map while generating an election probability map $p_{\max}$ (Figure 3.3c) that is directly proportional to the color variation in the original image. k coordinates are then randomly drawn according to the probability distribution described in p and readjusted to their original domain before pooling, resulting in k patches probabilistically containing highly variant and detailed sections of the painting (Figure 3.3).



(a) "Holländische Kanallandschaft" by Paul Baum, 1887.

(b) Canny filter's output for the painting.

(c) The $p_{\max}$ map for Baum's painting.

Figure 3.3: Examples of patches extracted from Baum's painting using the *max-grad* strategy.

3.1.1.3 Min-grad

Going in the opposite direction, we set to extract smooth sections of the painting, containing less corrections that mask one's brushstroke signature [26], and possibly modified by fewer adjacent hands (e.g., students).

This is done similarly to ımax-grad, but using $1 - p_{\max}$ as probability distribution, in the original painting being more likely to be randomly drawn. Figure 3.4 illustrates some examples of patches that were drawn following this policy.

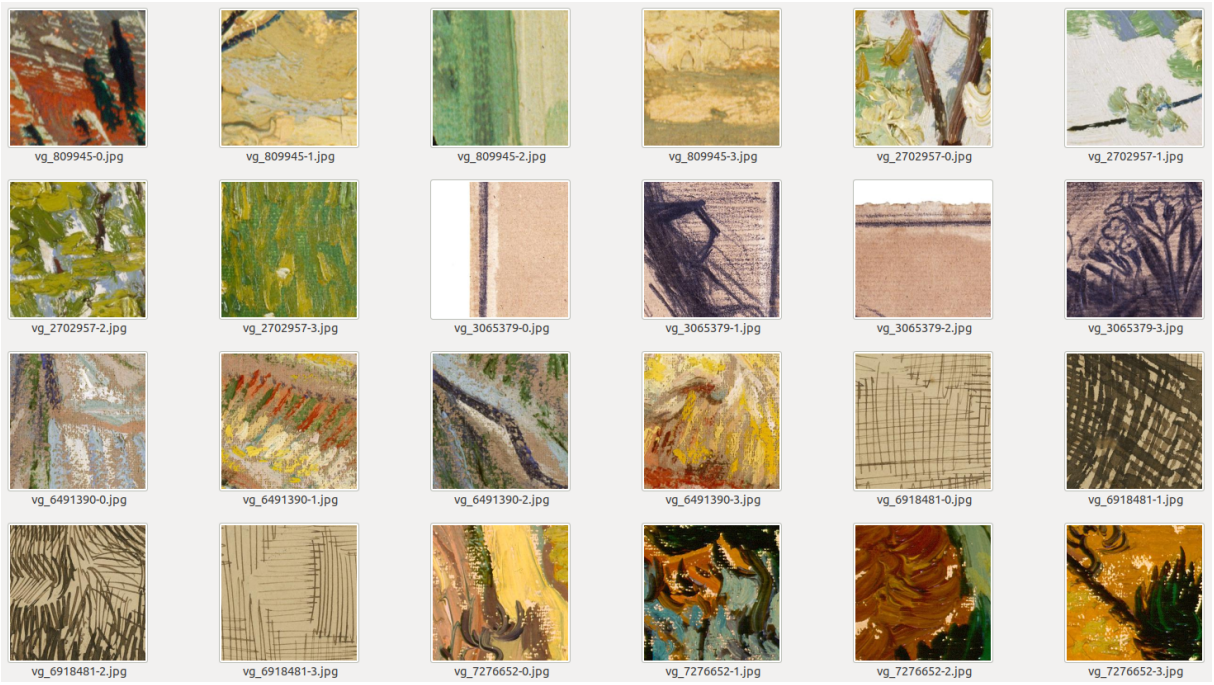


Figure 3.4: Samples of patches extracted with the min-grad strategy.

3.1.2 Transfer Learning from imagenet

We postulate that convolutional networks that have highly generic first layers capable of learning basic elements such as brushstrokes, and author-specific end layers that learn an artist's specific style can approximate, in accuracy, to art connoisseurs in authorship attribution, as well as art fraud detection.

Given all previously successful examples of transfer learning and the relatively small cardinality of vgdb2016, we first attemped to extract features using pre-trained frozen models and *warm-start* training with previously trained weights, optimized for the object classification onto imagenet [53]. During the evaluation phase, k patches of a painting were fed to the models, and the predictions are combined using either the strategies suggested in [31], or adaptations of them to the multi-class scenario.

The decision models tested in this project are described below. They were developed and tested using the scikit-learn [1] and Keras [54] libraries. The first subsection describes a simpler baseline method, based on color statistics, for comparison purposes.

3.1.2.1 Color Histograms *(baseline)*

Each patch of each painting is loaded and converted to HSV. Color histograms with 64 bins are computed from the **value** channel to create 64 feature vectors that are fed to a *Random Forest* binary classifier. Grid search is employed on the training of the top classifier, utilizing the parameters in Table 3.1. The answers for all patches in a painting are then aggregated to generate a final decision for the painting. Algorithm 1 in Appendix section generally lists the histogramming process employed here.

	values searched			
class-weight balancing	none	'balanced'		
RF Trees	10	50	100	1000

Table 3.1: Hyper-parameters grid-searched during training.

3.1.2.2 InceptionV3

The entire convolutional pipeline within InceptionV3 is reused, as well as its pre-trained weights over imagenet. *Global average pooling 2D* is applied to the network's output to permit a variable input size and a **fully-connected** layer with 2 units, **softmax** activation is attached to its end. Algorithm 2 describes the model in more detail. During training, the model is fine-tuned over $299 \times 299 \times 3$ patches from paintings in the van Gogh 2016 dataset using the *Adam* [55] optimizer. One third of the paintings are kept for validation. Training is conducted over n epochs and the weights that minimize the validation loss function are stored as test candidate.

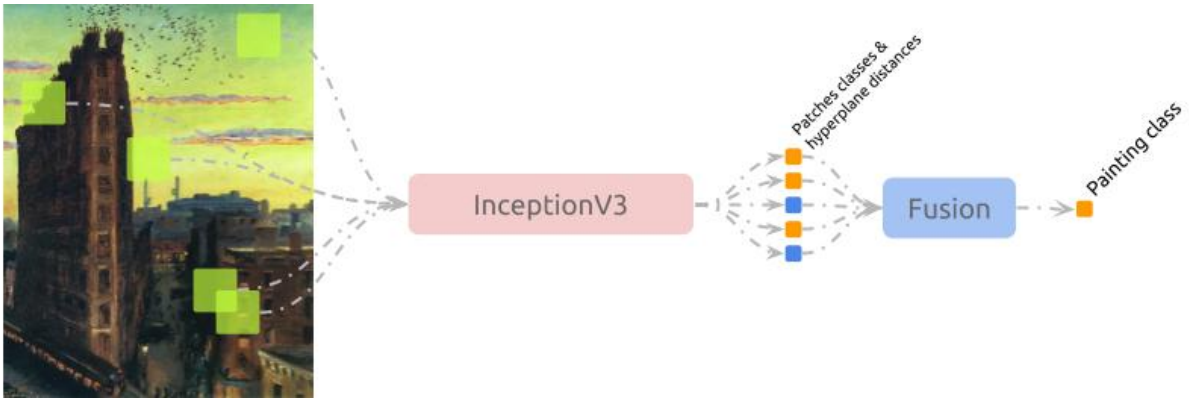


Figure 3.5: An illustration of the InceptionV3 authorship detection model.

3.1.2.3 InceptionV3+SVM

InceptionV3's conv. layers trained over imagenet are used *as-is* to extract a 2048 dimension feature vector from each painting patch. The generated dataset is used to train a *PCA + Support Vector Machine* binary classifier. The latter has its hyper-parameters tuned via the grid search method, varying conventional parameters such as regularizing factor C and kernel function (Table 3.2). For more details on the implementation of this model, see Algorithm 3.

	values searched				
PCA variance retained	95%	99%			
class-weight balancing	none	'balanced'			
SVC Kernel	'linear'	'rbf'			
SVC gamma	'auto' ($\frac{1}{1462}$)				
SVC C	.1	1	10	100	1000

Table 3.2: Hyper-parameters grid-searched during training. Every parameter not specified on the grid is left untouched, with the default value of the scikit-learn library [1].

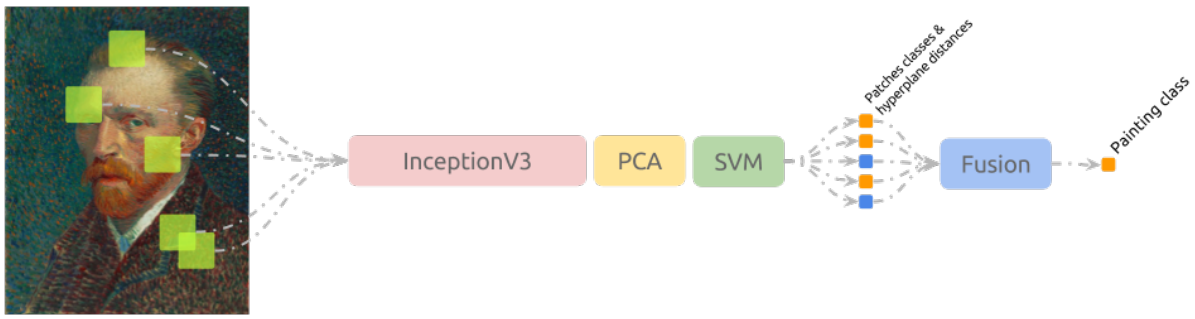


Figure 3.6: An illustration of the InceptionV3+SVM authorship detection model.

3.1.2.4 DenseNet 264

While 299 pixel patches would make the artist's brushstrokes clear for most large paintings, it could still retain full semantic items such as trees and houses for the smaller images.

Concerned that this could add some bias to the discriminator, we have also tested with a much smaller patch size and network size. In this sense, 500 patches of shape $32 \times 32 \times 3$ are extracted from the van Gogh 2016 dataset and fed to a DenseNet 264 network [56] attached to a *softmax* classifier, trained from scratch.

3.1.2.5 DenseNet 264+SVM

The model described in Section 3.1.2.4 is clipped at its final *global average pooling 2D* [57] and used to extract features from the dataset. All patches are transformed with it and fed to a *PCA + Support Vector Machine* binary classifier. Grid search is once again employed to train the top classifier, using the same parameters presented in Table 3.2.

3.1.2.6 Contrastive VGG19

The VGG19 convolutional pipeline (with its weights trained over imagenet) is used as embedding network L, which is duplicated and jointed with the l^2 distance function (see Algorithm 4 for further details). The entire model is then trained with *contrastive loss* on the van Gogh 2016 dataset. This induces the model to embed patch pairs that belong to a single artist close together, while projecting distinct authored patches further apart.

During the evaluation phase, p patches for each painting of unknown authorship are paired with p random patches extracted from van Gogh's work present in the training set. The distances found are averaged into a single one, and authorship matching is confirmed if said distance does not overreach a given threshold (empirically established from the validation data).

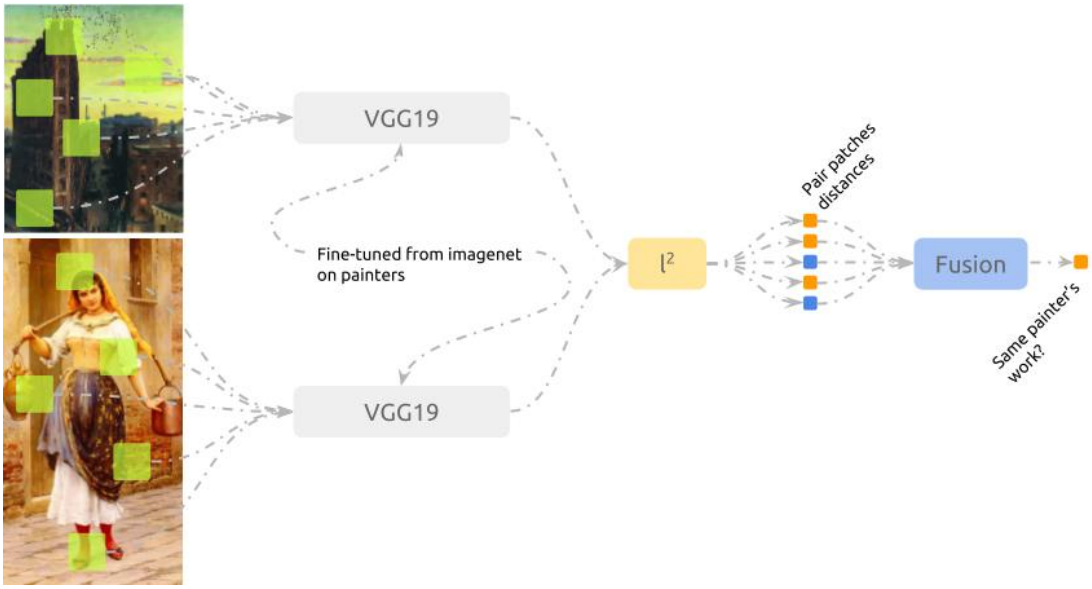


Figure 3.7: An illustration of the Contrastive VGG19 authorship detection model.

3.1.3 Cross-Dataset Testing

In addition to the test set defined within vgdb2016, we attempted to further validate the trained models over recaptures (of the ones present in vgdb2016 test set) and unseen

(a) Sample 9443864 in vgdb2016. (b) Recapture from the museum.

Figure 3.8: “A way with the entrance to a viewpoint" by Vincent van Gogh, 1887.

samples authored by artists present in vgdb2016. By testing the models over samples created under different capturing conditions, with various resolutions, pixel density and sources, we were able to further demonstrate the extensibility of the models in a more realistic scenario, where one cannot ensure optimal capturing settings. Three datasets were collected.

3.1.3.1 Recaptures from the van Gogh Museum

Recaptures of paintings in the van Gogh 2016 test set were retrieved from the van Gogh museum website [58]. Due to restrictions in the site's search engine and translation difficulties, merely eight van Gogh recaptures were found, identified in the vgdb2016 by the following codes: 9106795, 9386980, 9387502, 9413420, 9414279, 9421984, 9443864 and 9506505. Nonetheless, this set can be used to *assert the accuracy of the trained models over high-quality copies of van Gogh paintings*. Figure 3.8 illustrates a painting from vgdb2016 and its recapture extracted from the van Gogh Museum.

3.1.3.2 Unseen Paintings from the van Gogh Museum

Images of paintings that do not appear in vgdb2016 were retrieved from the vangogh museum website. This set is composed by 34 van Gogh paintings and 11 paintings by other artists somewhat related to van Gogh. An interesting question could be answered here: *how well does a model trained over vgdb2016 generalize for unseen van gogh paintings?* Figure 3.9a and Figure 3.9b illustrate two paintings of different labels from this set.

3.1.3.3 Recaptures from Multiple Places

Recaptures of paintings in vgdb2016 test set were retrieved from multiple locations, found using google image search. Being composed by 55 van Gogh recaptures and 85 recaptures

(a) “Sunflowers" by Vincent van Gogh, 1889.

(b) “Tulip Fields near The Hague" by Claude Monet, 1886.

Figure 3.9: Paintings unseen by the trained models, extracted from the van Gogh Musem.

from other artists, this set is ideal for *understanding the overall performance of the models on an unrestricted scenario*. Figure 3.10 ilustrates multiple paintings from a single painting from vgdb2016 test subset.

While gathering this set, images with closer resolutions to the ones presented in van Gogh 2016 dataset were given preference. Unfortunately, while vgdb2016 contains large files, recaptures from Google were frequently smaller than 2000 per 2000 pixels. In an attempt to account for this, two sets of evaluation were performed:

1. No preprocessing was done to the images, and the models were tested over them as they were when retrieved.

2. The recaptures were resized so their largest dimension (i.e., height or width) would match their counterpart in vgdb2016. Unseen paintings were clustered into “vertical" and “horizontal", and resized so their largest dimension would match with the average size of paintings in vgdb2016 test set. All resizing performed preserved the image's original aspect ratio.

group	**height**	**width**	**area**
minimum	1,016 px	1,044 px	1,419,840 px^2
average	3,115 px	2,805 px	10,047,495 px^2
maximum	7,172 px	7,243 px	45,355,666 px^2

Table 3.3: General statistics for the test images in van gogh 2016 dataset.

3.1.4 Combining Recaptures

Following the idea of sample ensemble, we hypothesize that combining the classification of multiple recaptures of a same painting can induce greater variability to the system and

(a) “Landscape with Wheelbarrow" by Vincent van Gogh, 1883.

(b) recapture #0

(c) recapture #1

(d) recapture #3

Figure 3.10: Paintings unseen by the trained models, extracted from the van Gogh Musem.

increase its overall confidence. To demonstrate this, the model described in Section 3.1.2 with best scoring was used to classify each patch of each painting recapture. The results of each painting were aggregated using the **frequent** strategy, yielding the final answer for a painting recapture group. The results obtained here could then be directly compared with the one obtained in 3.1.3.3.

Additionally, the vgdb2016 test set is merged with their random recaptures to attempt to increase accuracy in vgdb2016. The results are then compared to the ones above.

3.1.5 Training With Recaptures

While Section 3.1.4 is important to assert the influence of recapture ensembling when testing, it is not sufficient to determine how much of error can be attributed to the lack of variance in the training set. To understand this, multiple recaptures of each painting in the vgdb2016 train set were collected from various sources – similarly to what was done in Section 3.1.3.3 –, and a new training set was formed by merging all training samples and their recaptures. The best scoring model described in Section 3.1.2 was re-trained over this new set and evaluated over vgdb2016 test set, *RMP* and the union of both.

3.2 Experiments and Results

This section presents the results of the experiments employed to demonstrate the claims made in Section 3.1. The subsections were enumerated to follow the same structure of the previous section.

3.2.1 Sampling Few Patches to Represent a Painting

Here, the influence of patch count decrease is reported over different models. Table 3.4 shows the accuracy scores obtained in validation, patch test (over the test patch samples) and test, after patch ensemble is performed.

Validation accuracy is unreported in all *vgg19* and *InceptionV3* based classifiers due to the fact that these networks were only used as feature extractors and did not engage in training phase in which a validation metric could be extracted. Furthermore, patch-level test accuracy was not reported in [31].

#	**classifer**	**patches**	**patch validation accuracy**	**patch test accuracy**	**painting test accuracy**
1	vgg19(fc2) SVM [31]	all (≈ 205)	-	?	94.03%
2	**vgg19(fc2) pca SVM**	50 random	-	**89.31%**	**95.52%**
3	vgg19(fc2) pca SVM	20 random	-	87.91%	91.04%
4	**vgg19(flatten) SVM**	50 random	-	**89.71%**	**95.52%**
5	vgg19(flatten) SVM	20 random	-	88.95%	91.04%
6	**InceptionV3 pca SVM**	50 random	-	**89.04%**	**95.52%**
7	**InceptionV3 pca SVM**	20 random	-	**89.70%**	**95.52%**
8	InceptionV3 pca SVM	10 random	-	88.96%	94.03%
9	InceptionV3 pca SVM	5 random	-	88.9%	92.54%
10	InceptionV3 pca SVM	2 random	-	86.99%	91.04%
11	InceptionV3 pca SVM	1 random	-	86.47%	83.58%
12	densenet softmax	500 random	82.42%	84.21%	92.54%
13	densenet softmax	200 random	87.08%	86.63%	88.06%
14	densenet softmax	50 random	87.08%	84.20%	92.54%
15	densenet pca SVM	50 random	87.08%	87.01%	91.04%

Table 3.4: Accuracy observed per model, when gradually decreasing the patches used in training.

3.2.2 Transfer Learning from Imagenet

Experiment results on the application of transfer learning to the van Gogh attribution problem are presented in this section. Table 3.5 lists the many models employed, the

patch extraction strategy/dimension, and the test accuracy score over vgdb2016 for both patches and paintings.

Any attempt to fine-tune last networks (such as InceptionV3) besides the last classifying layer resulted in overfitting and, later, test accuracy decrease. This is due to the large number of parameters/active units in contrast with the insufficient data set size. This problem was less apparent when training the DenseNet model, as each painting in the train/test set was sub-divided into roughly 500 patches and short training was performed.

Finally, we observed regularizing strategies — such as Dropout — playing a key role in the training of models 9, 11, 13 and 14, even though these have been outperformed by the SVM due to the stratified K-Fold cross-validation strategy employed during its training. Test accuracy values present little to no difference, indicating this set is too small for any deeper comparison between the strategies used.

#	**classifer**	**patches**	**patch test accuracy**	**painting test accuracy**
1	vgg19(fc2) SVM [31]	all (≈ 205) 224px	?	94.03%
2	**vgg19(fc2) SVM**	**random 224px**	**89.31%**	**95.52%**
3	histogram 64 bins Random Forest	random 299 px	73.19%	79.10%
4	InceptionV3(mixed4) SVM	min-grad 299 px	90.75%	94.03%
5	InceptionV3(mixed7) SVM	min-grad 299 px	90.42%	94.03%
6	**InceptionV3 PCA SVM**	**min-grad 299 px**	**88.69%**	**95.52%**
7	InceptionV3 PCA SVM	max-grad 299 px	88.72%	92.54%
8	**InceptionV3 PCA SVM**	**random 299 px**	**90.99%**	**95.52%**
9	fine-tuned InceptionV3 PCA SVM	min-grad 299 px	90.69%	94.03%
10	Xception pca(0.95) SVM	random 299 px	89.03%	94.03%
12	ResNet50 pca(0.99) SVM	random 299 px	88.88%	91.04%
11	Contrastive VGG19	random 299 px	-	79.10%
13	densenet-264 softmax	random 32 px	84.21%	92.54%
14	densenet-264 pca(0.95) SVM	random 32 px	87.01%	91.04%

Table 3.5: Accuracy observed per strategy/approach.

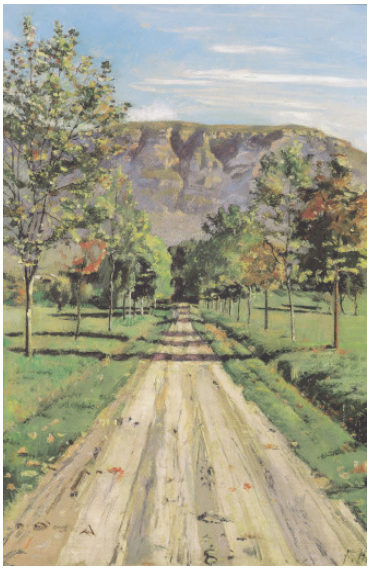
	nvg	vg
nvg	87%	13%
vg	5%	95%

Table 3.6: patch-level confusion matrix of model *#8* in Table 3.5, over van Gogh 2016 test set.

	nvg	vg
nvg	93%	7%
vg	0%	100%

Table 3.7: painting-level confusion matrix of model *#8* in Table 3.5, over van Gogh 2016 test set.

(a) “Mädchenbildnis", Paula Modersohn-Becker, 1905.



(b) “Die Strasse nach Evordes", Ferdinand Hodler, 1890.

(c) “The Palazzo Contarini, Venice", Claude Monet, 1908.

Figure 3.11: Paintings of the van Gogh 2016 test set that were misclassified by model #8 in Table 3.5.

3.2.3 Cross Dataset Testing

3.2.3.1 Recaptures from the van Gogh Museum

Table 3.8 presents results over the eight van Gogh recaptures from the van Gogh Museum. Only the true positive rate is shown, considering only van Gogh paintings compose this set.

#	classifier	patches	pre	best st	patch tpr	tpr
1	**vgg19(fc2) [31]**	**all**	**none**	**far**	**11%**	**25%**
2	**InceptionV3 SVM**	**random**	**none**	**freq**	**28%**	**25%**
3	InceptionV3 SVM	random	resized	m, far, fq	24%	13%
4	InceptionV3 SVM	min-grad	none	m, far, fq	27%	13%
5	InceptionV3 SVM	min-grad	resized	m, far	21%	13%
6	densenet264	random	none	far	02%	0%
7	densenet264	random	resized	far	07%	13%

Table 3.8: evaluation results for some of the models defined in Section 3.1.2 over the recaptures from the van Gogh Museum.

While the column names were abbreviated for visualization purposes, they can be briefly described as follows. **pre.** lists the pre-processing method applied to patches. **best st.** describes the strategy resulting of the highest accuracy score. The **support** of vgg19(fc2) [31] is 1231 patches, while all other results are supported by exactly 400 patches. **Patch tpr** describes the patch-level true-positive rate, while **tpr** lists

the painting-level true-positive rate, after ensemble is performed. Finally, **acc.** is the painting-level class-balanced accuracy score of the model.

Figure 3.12: Misclassifications in the recaptures from the van Gogh museum data set.

3.2.3.2 Unseen Paintings from the van Gogh Museum

Table 3.9 presents the results over unseen paintings from the van Gogh Museum, considering multiple models, patch extraction methods and pre-processing functions. Its columns contain, besides the attributes from Table 3.8, the patch true positive and negative rates (**ptpr**, **ptnr**) and sample (painting) level true positive and negative rates. Once again, vgg19(fc2) [31]'s results support differs from the others, being 8,623. The remaining results are supported by 2,250 patches.

classifier	patches	pre.	best st.	ptpr	ptrn	tpr	tnr	acc.
vgg19(fc2) [31]	all	none	far	86%	20%	82%	29%	56%
InceptionV3 SVM	random	none	m, far	86%	45%	91%	41%	66%
InceptionV3 SVM	**random**	**resized**	**far**	**85%**	**42%**	**100%**	**44%**	**72%**
InceptionV3 SVM	min-grad	none	far	85%	45%	100%	41%	71%
InceptionV3 SVM	min-grad	resized	far	86%	44%	91%	44%	68%
densenet264	random	none	far	86%	11%	73%	24%	48%
densenet264	random	resized	far	90%	10%	91%	24%	57%

Table 3.9: Evaluation results for some of the models defined in Section 3.1.2 over the unseen paintings from the van Gogh Museum.

3.2.3.3 Recaptures from Multiple Places (RMP)

Paintings from this set were grouped by painting, and the “frequent" strategy was used to form a final classification for the painting. Table 3.10 reports the application of multiple classification strategies onto this set. 31,427 patches support vgg19(fc2) [31]'s results, while the resutls of the other models are built comprising of 7,100.

model	patches	pre.	st.	ptpr	ptnr	tpr	tnr	acc.
vgg19(fc2) [31]	all	none	far	80%	56%	93%	63%	78%
InceptionV3 SVM	**random**	**none**	**fq**	**91%**	**66%**	**96%**	**67%**	**82%**
InceptionV3 SVM	random	resized	far	87%	68%	92%	67%	78%
InceptionV3 SVM	**min-grad**	**none**	**m, fq**	**90%**	**67%**	**96%**	**67%**	**82%**
InceptionV3 SVM	min-grad	resized	fq	86%	68%	93%	67%	80%
DenseNet264	random	none	mean	90%	34%	95%	30%	63%
DenseNet264	random	resized	far	92%	34%	93%	40%	67%

Table 3.10: Evaluation results for some of the models defined in Section 3.1.2 over *recaptures from multiple places*.

3.2.4 Combining Recaptures

The recapture-level classification results ensemble performed over samples in *recaptures from multiple places* is reported in Table 3.11, resulting in a class-balanced accuracy score of 87%. The following samples were incorrectly classified: 9103139, 9386980, 9387502, 9414279, 9421984, 9463012 and 10500055.

A class-balanced accuracy of 93% was achieved when testing the merge between *van Gogh 2016 test set* and *recaptures from multiple places* (Table 3.12). The miss-classified samples were now 9103139, 9414279, 10500055, 10658644 and 18195595.

	nvg	vg
nvg	97%	3%
vg	24%	76%

Table 3.11: painting-level confusion matrix of 38 non-van Gogh and 25 van Gogh recapture groups from *RMP*.

	nvg	vg
nvg	93%	7%
vg	8%	92%

Table 3.12: painting-level confusion matrix of 42 non-van Gogh and 25 van Gogh recapture groups from *RMP* ∪ vgdb2016 test set.

3.2.5 Training With Recaptures

3.2.5.1 van Gogh 2016 Test Set

A patch-level class-balanced accuracy score of 91% over the van Gogh 2016 test set was reached with a classifier trained over van Gogh 2016 train set and their recaptures from multiple places (Table 3.13). The painting-level class-balanced accuracy score of 96% was achieved after patch ensemble was performed, in which the misclassified samples were 10500055, 10658644 and 18195595 (Table 3.14).

	nvg	vg
nvg	86%	14%
vg	3%	97%

Table 3.13: patch-level confusion matrix over 3350 patches in the van Gogh test set.

	nvg	vg
nvg	93%	7%
vg	0%	100%

Table 3.14: painting-level confusion matrix over 67 paintings in the van Gogh test set.

3.2.5.2 Recaptures from Multiple Places

After training a classifier over samples in the van Gogh 2016 train set and their recaptures from multiple places, a patch-level class-balanced accuracy of 79% was achieved, comprehending 5737 correctly classified patches over a total of 7100 (Table 3.15). The recapture-level accuracy reached 83% after patch-ensemble was performed (Table 3.14) and, finally, 87% after recapture ensemble (Table 3.17).

	nvg	vg
nvg	89%	11%
vg	32%	68%

Table 3.15: patch-level confusion matrix over 7100 patches in the RMP set.

	nvg	vg
nvg	95%	5%
vg	30%	70%

Table 3.16: recapture-level confusion matrix over 142 recaptures in RMP set.

	nvg	vg
nvg	97%	3%
vg	24%	76%

Table 3.17: painting-level confusion matrix over 67 recapture groups in RMP set.

	nvg	vg
nvg	95%	5%
vg	4%	96%

Table 3.18: painting-level confusion matrix over 67 recapture groups from $RMP \cup$ vgdb2016 test set.

The following recaptures from the RMP set were incorrectly classified: 18195595-0, 9463012-0, 9386980-1, 9414279-2, 9413420-0, 9413420-1, 9103139-2, 9103139-1, 10500055-0, 9110201-1, 9387502-1, 9414279-0, 9387502-3, 9103139-0, 9414279-1, 9106795-1, 10500055-1, 9386980-0, 9780042-2, 9378884-3, and 9103139-3.

After recapture-ensemble was performed, the following paintings were misclassified: 10500055, 9103139, 9386980, 9387502, 9413420, 9414279, and 9463012.

When considering both RMP and vgdb2016 set as one, three paintings were incorrectly labeled: 10500055, 10658644, and 9414279.

3.3 Discussions

3.3.1 Sampling Few Patches to Represent a Painting

No accuracy decrease is observed in Table 3.4 until patch count is down to 10, indicating fewer patches are sufficient to represent a painting. Performance, on the other hand, increased significantly as we reduce patch count: model #1 - vgg19(fc2) SVM [31] took 5 minutes and 36 seconds to embed all patches (using the maximum allowed batch size of 158), and approximately three hours to grid-search the top SVC's parameters (*rbf kernel*, $C = 0.1$) and effectively train it. Meanwhile, model #6 has only taken 2 minutes, 15 seconds to embed (batch size of 362), and 18 minutes to train (where the best parameters found are the *rbf kernel* and $C = 10$). Evaluation time for both classifiers was negligible.

3.3.2 Transfer Learning from Imagenet

Combining the InceptionV3 convolutional pipeline with the Support Vector Machine generated the best accuracy score over patches in vgdb2016 test set: 91%. The confusion matrix presented in Table 3.6 shows a more detailed view of the performance of this model.

Combining patches increased the overall accuracy and achieved perfect discernment for van Gogh paintings, while missing three non-van Gogh paintings. This setting is shown in Table 3.7, and it is best summarized by an observed class-balanced test accuracy score of 96.5%.

However, three non-van Gogh paintings (10500055, 18195595 and 10658644) were incorrectly mistaken for van Gogh's work (Figure 3.11). As van Gogh, the painters of the first two – Paula Modersohn-Becker and Claude Monet – are closely related to impressionism. The last painting comprehends a landscape, which is often subject of van Gogh's work.

3.3.3 Cross Dataset Testing

3.3.3.1 Recaptures from the van Gogh Museum

From Table 3.8, we conclude that none of the models were able to generalize vgdb2016 to this set. From all eight paintings, only two – 9443864 and 9506505 – were correctly classified into the *van-gogh* class. The other six misclassified paintings are shown in Figure 3.12. Furthermore, half of the misclassified samples are also commonly misclassified in Recaptures from Multiple Places (see Section 3.3.3.3 for more details), leading us to believe this subset simply contains the most difficult samples of the original set.

A series of hypotheses to explain the negative results were considered, such as strong differences in format, encoding, quality and/or size. Such features were slightly altered in order to approximate to vgdb2016 samples, but results did not improve. Finally, we were left with the suspicion that scanning these samples instead of photographing them has ultimately induced a significant difference between the features in this set and the ones in vgdb2016.

However, it is troublesome to make assumptions onto a dataset with only eight samples, as results might be highly biased with high changes of noise appearance. Still, these results were reported for comprehensiveness and academic purposes.

3.3.3.2 Unseen Paintings from the van Gogh Museum

We observe the model slightly overfitted (onto vgdb2016 train/test), as the “non-van Gogh" label becomes more likely to be predicted. However, it was still able to maintain some degree of certainty. For the best decision strategy (InceptionV3 SVM, using resized paintings and random patches), the model was able to achieve a 72% class-balanced accuracy score, in which the misclassifications are van Gogh's work. This demonstrates that the trained models are indeed able to generalize some of the information from the photographed paintings to the scanned, high-quality ones.

3.3.3.3 Recaptures from Multiple Places

Our best classifier achieved the highest score on this set, which is likely due to the fact that these images were photographed, in opposite of scanned. Table 3.15 shows the model scored 79% of patch-level and 82% of recapture-level class-balanced accuracy (12% and 14%, respectively, less than the same metric over vgdb2016 test set presented in Table 3.6). We conclude from these numbers the model effectively generalized the patterns within the data.

3.3.4 Combining Recaptures

We combined the classifications of multiple recaptures into a painting-level answer, which resulted in a painting-level balanced accuracy of 87% — five percent points more than the score achieved when not merging them —. This indicates that combining the classifications of multiple captures of a given painting can further increase the thrust of a model when deciding upon it.

If vgdb2016 test samples are among the ones combined, then the same metric goes to 93%. Therefore, the recaptures have decreased the decision pipeline's overall accuracy (by making more incorrect classifications or returning less confident answers, closer to the decision hyperplane).

3.3.5 Training With Recaptures

3.3.5.1 van Gogh 2016 Test Set

Patch-level class-balanced accuracy is 1% above the one reached when trained over only the original van Gogh set. Painting-level accuracy is the same as the original one and the missing samples the same as in Figure 3.11. While we cannot state adding multiple recaptures to the training will help to mitigate the error of the decision pipeline, it has not decreased its efficacy either.

3.3.5.2 Recaptures from Multiple Places

Patch-level accuracy stays at 79%, just as its counter-part trained exclusively in the original van Gogh set. The difference between the probability of correct guesses to the positive and negative classes has, however, decreased by one percent point; indicating a slight reduction in specificity.

Recapture-level accuracy was improved by two percent points, showing that the increase variation in training data has a positive impact on the score of our decision method.

Painting-level accuracy did not show any change from the original training. However, an improvement of three percent points is seen when adding the samples from the original set to the ensemble. This is a sign the model is more confident when classifying the recaptures, which in turn generate more stable ensembles, and a better final decision function.

Chapter 4

Provenance Analysis for Multiple Painters

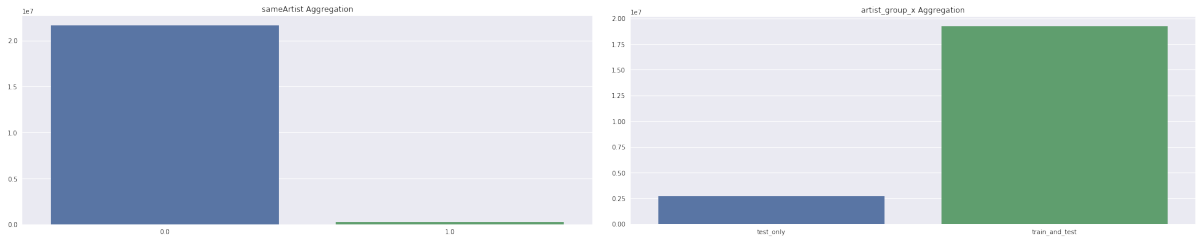
An immediate extension of the work performed in Chapter 3 is to estimate provenance in a multi-class scenario, whereby the collected features do not necessarily reflect the work of a single painter, but a broad range of artists relating to distinguishing styles and background. In this same scope, it is reasonable to assume artists and styles, which were unknown to the model in training will eventually emerge during the test phase. Therefore, it is desirable for a machine-learning model to not only be able to directly classify authorship, but to discriminate art patterns on a open-class scenario.

Multiple issues arise when dealing with such a diverse setting. The expected performance and accuracy figures will usually decrease, as more complicated decision functions are required to better discriminate classes; class unbalance and the curse of dimensionality become more prominent issues, difficult to circumvent. Finally, testing becomes more convoluted. We opted to start our study over an already established art dataset: Painters by Numbers, granting an easier way to perform comparison with literature.

The Painter by Numbers dataset [59] — a subset of the Wikiart [22] made available in a Kaggle competition of same name — was used to train and evaluate the efficacy of our models. In the training set, each one of its 79,432 paintings is labeled among the 1584 painters, 135 styles, 42 genres. An approximate creation year is also available, though some entries are missing. The set is highly unbalanced among all of its label groups (Figures 4.2 and 4.3). Its test sub-set contains 13 major groups of paintings, with a total of 23,817 samples that either have or not appeared in the training set (Figure 4.1b). The associated challenge in the Kaggle competition comprises 21,916,047 test cases, each containing a pair of paintings. Contestants were expected to answer each test case with the labels 0, if the pair is believed to be from different authors, or 1, if they share authorship. As the great majority of test pairs are not from the same painter, the test set is also highly unbalanced, as Figure 4.1a shows. To overcome the incorrect reading from models overfitted over the group with highest cardinality, ROC AUC was used as scoring evaluating function.

In this Chapter, we adapted the strategy used in the previous chapter to a multi-class scenario. Combining authorship, style and genre information, we create a data-driven model capable of competitive accuracy, performance and explainability over the Painter

by Numbers dataset.



(a) Proportion of painting pairs with different authors (0.0) for pairs with same artist (1.0).

(b) Fraction of test pairs, whose paintings appear in test or in both training and test sets.

Figure 4.1: Binary aggregations for samples in the training and test set.

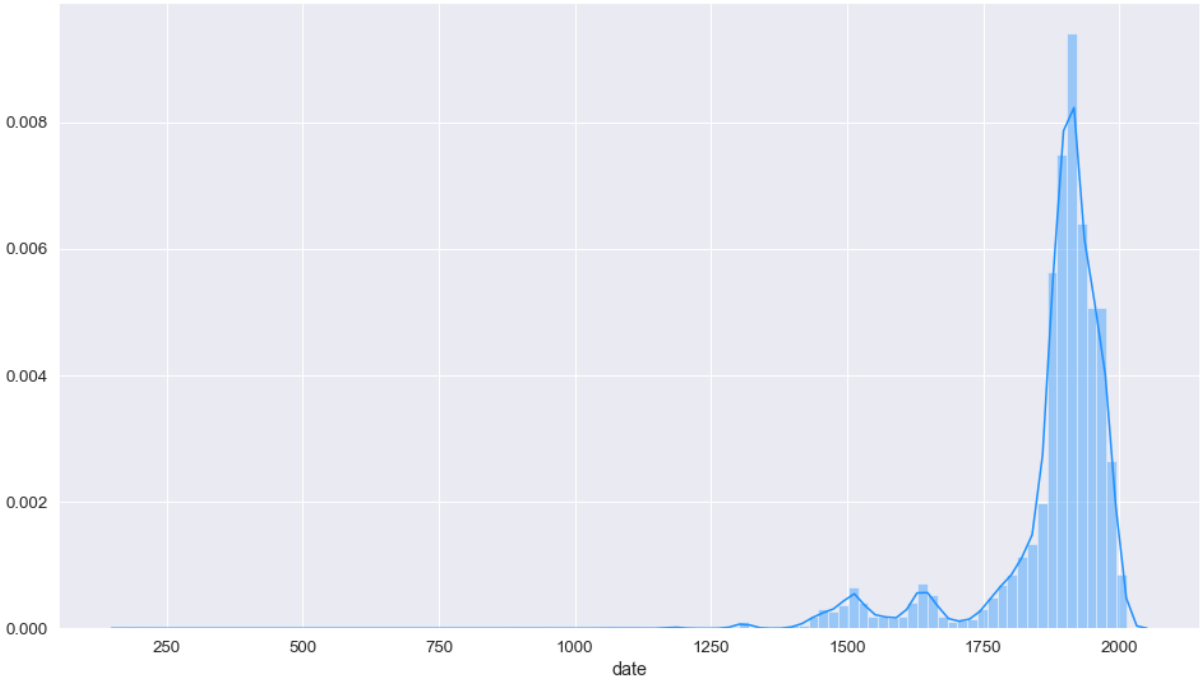
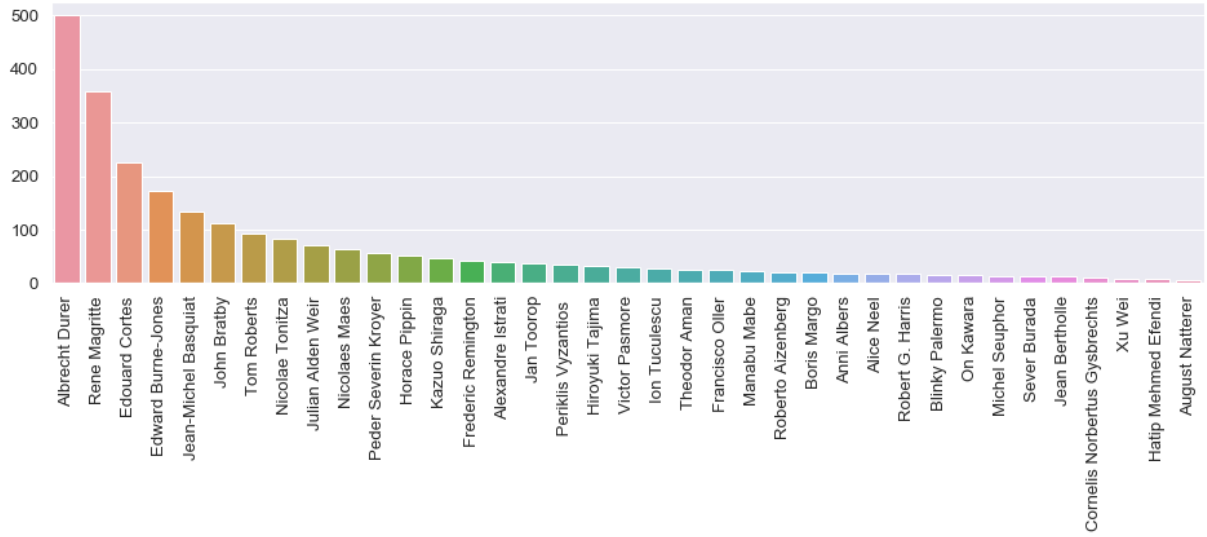
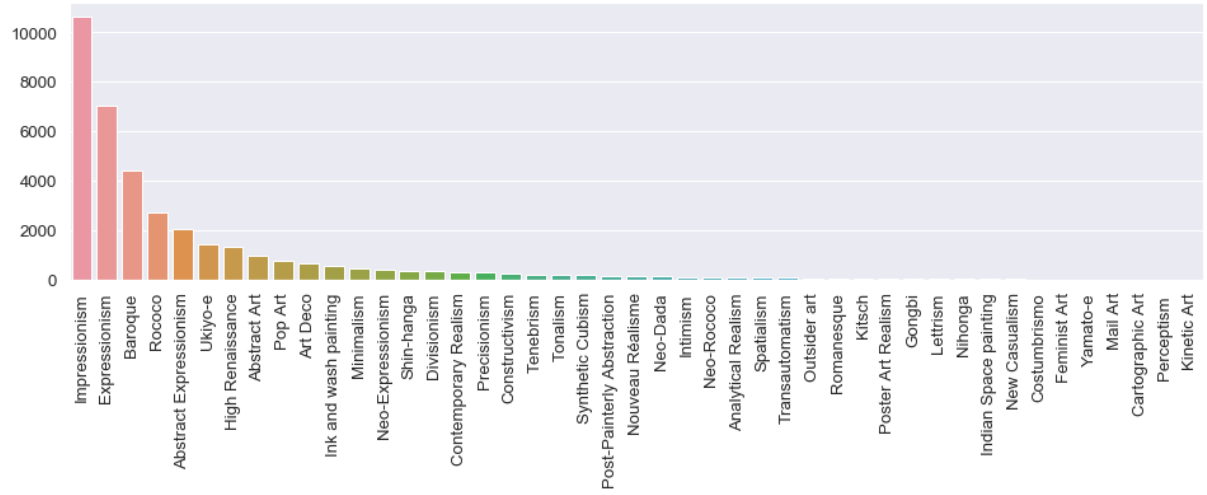


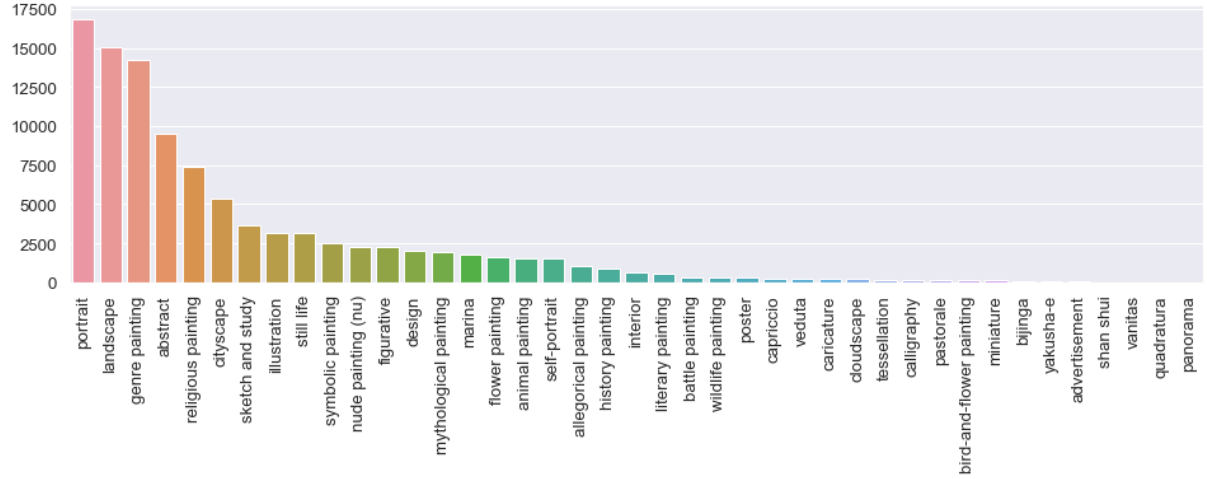
Figure 4.2: Histogram of paintings by their creation year.



(a) Number of paintings per artist label. Only a short subset is shown for briefness. Examples were uniformly selected from the all painters sorted by their occurrence frequency.



(b) Number of paintings per style label. Only a short subset is shown for briefness. Examples were uniformly selected from the all styles sorted by their occurrence frequency.



(c) Number of paintings per genre label.

Figure 4.3: Sample occurrences per class observed (painter, style and genre). Labels containing fewer than three samples were omitted in each group.

4.1 Proposed Methodology

We discuss in the following sections our method for solving Painter-by-Numbers decision task. We kept the main premises employed when studying van Gogh's work. Our method comprises three main steps: training an embedding network L capable of transforming image patches into a semantically rich, more organized space. Training a binary network that takes two input patch streams and discriminates between matching and non-matching authorship and, finally, combining with some of the ensemble strategies described in Section 3.1.4.

4.1.1 Normalizing Inconsistent Sample Frequencies

Most paintings in Painters by Numbers were collected from the WikiArt set, which in turn was created from multiple sources. Samples from both train and test sets vary greatly in size. These conditions are far from ideal, as they create an unpredictable setting, in which the number of times two different paintings can be sliced without repetition differs considerably. Under-sampling patches would be equivalently inadequate, given many painters are matched with very few samples, resulting in a severe loss of train data.

We considered the two following alternatives in order to mitigate this sample frequency inconsistency:

1. Fifty $299 \times 299 \times 3$ patches are extracted from each painting in the train and validation sets, yielding 3,971,700 samples. In the worse (and yet unrealistic) case scenario, a painting is as small as the desired patch size target and it is sliced into 50 equal patches. On the other hand, large paintings are likely sliced into very different ones. By defining a batch size of 64 samples, at least 869 steps would be necessary for each painting to have at least one of its patches processed. We set an epoch to comprise 869 steps, with batches of 64 patches — sampling 55616 patches in each epoch —, which will unlikely include too many repetitions for a single painting.

2. A fixed amount of patches is set (e.g. 1,000). All possible patches are extracted from a given artist's train paintings and 1,000 of those are sampled (with repetition) from the set. This achieves a balanced training set with a probable low number of repetitions. The epoch comprises all training patches available.

In both alternatives, small random variations are further applied to each patch before being fed to the network during training, including horizontal and vertical flipping, brightness shifting and resizing (up to 20% increase or decrease of the original size).

4.1.2 Partitioning Networks for Performance and Memory Concerns

Architectures such as InceptionV3 and InceptionResNetV2 contain a massive number of layers and units, entailing a considerable impact on memory and performance. Even

when training simple non-siamese networks, common batch sizes are in between 256 and 512. Moreover, duplicating a feed-forward in order to generate two limbs would quickly consume all memory available of the computation host devices, as two matrices groups are required to store the signal feed-forward through each one of the limbs.

During test, model 4.1.3.7 could only support a batch-size of 48 when running on a GeForce Titan X graphic card with 12 GB of RAM. Considering the 21,916,047 different test cases, we estimated that approximately three weeks would be necessary to finish processing the entire test set.

In order to circumvent this issue, we separated our models into different segments, fitting each segment separately. After each training, all samples in the dataset were feed-forward and the produced output was saved into disk so it could be loaded for the next step. Segments were chosen such that significant processing was done in each step, while still allowing a reasonable batch size. Figure 4.4 displays the partition, where the feed-forward signal is saved into disk and its sections are trained separately. Grey areas represent the extent of gradient back-propagation in each block's training.

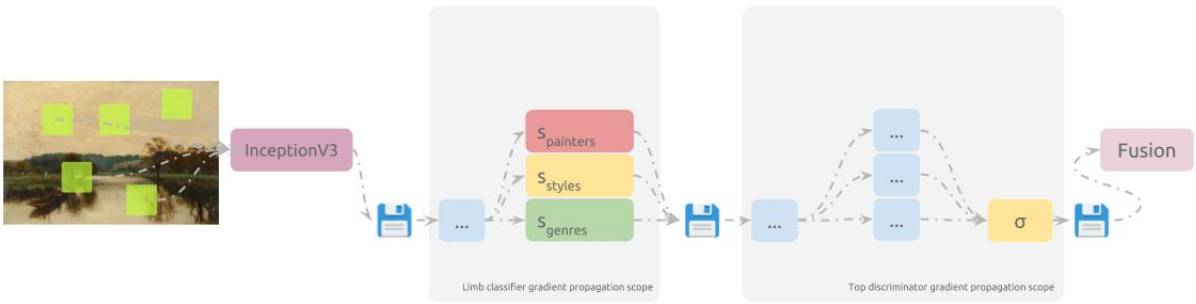


Figure 4.4: Diagram of a parted network's training.

See each Subsection 4.1.3 for more details on how each model was parted.

4.1.3 Strategies and Decision Models

The sections below describe the networks used in the attempt of solving the Painter-by-Numbers competition. Slight variations might have been tested and are listed in the Section 4.2. Following what was found in the previous chapter, only 50 randomly extracted patches are used to represent a single test painting. No attempt in balancing the dataset — besides weighting the sample gradients by the inverse of their class frequencies — is attempted unless specified otherwise.

4.1.3.1 InceptionV3, PCA, SVC and Equal Joint

This strategy resembles model 3 in Section 3.1.2.3 — used to discriminate van Gogh's work — as the last (1000 units) dense and (softmax) activation layers are removed from the InceptionV3 model pre-trained over imagenet. The remaining pipeline is used to extract

features from 100 random painters in the train and validation sets[1], which are fed to a PCA/SVM classifier. Table 4.1 lists the hyper-parameters grid-searched used to train the top model.

Once the classifier is trained, the 50 patches of each test painting are presented to it and a label is generated for each painting using the *frequent* strategy. A given test case containing two paintings (a, b) is answered with 1 if the label predicted to a is the same as the one for b, and 0 otherwise. As a side note, this decision strategy does not expose a probability or distance which can be interpreted as decision confidence and, therefore, does not allow the computation of the ROC AUC metric. Figure 4.5 illustrates the discrimination process explained above.

	values searched				
class-weight balancing	None	"balanced"			
SVC Kernel	"linear"	"rbf"			
SVC C	.1	1	10	100	1000

Table 4.1: Hyper-parameters grid-searched during training.

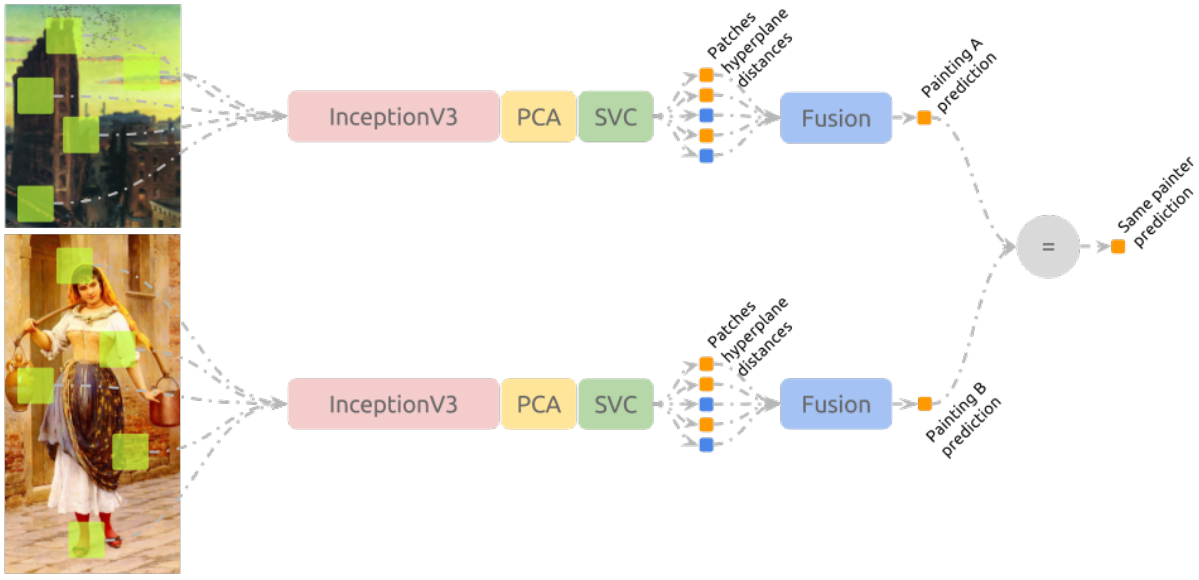


Figure 4.5: Diagram of the InceptionV3, PCA and SVC classification pipeline jointed by the equal operation.

4.1.3.2 Siamese InceptionV3, Dot Joint

Painting patches are used to fine-tune a feed-forward network L, composed of the convolutional layers of InceptionV3 and a final dense layer with 1584 units and *softmax*

[1]The selection process consisted of sorting painters by their random identifying code and selecting the 100 first ones.

activation in order to classify its painter. The weights are again initialized with the ones that optimize classification in the imagenet dataset.

Once L is trained, each sample (a group of 50 patches) in the test set is fed to the network in order to produce their 50 prediction vectors, which are saved into disk. The ensemble strategy *mean* is used to merge all of the vectors into a single probability distribution vector.

Assuming two paintings of the same painter will contain similar prediction vectors, we can use the inner product of the two resulting prediction vectors to obtain the correlation between the paintings. If two samples $\boldsymbol{x}$ and $\boldsymbol{y}$ are correlated ($p_{\boldsymbol{x},\boldsymbol{y}} > 0$), they are considered a match, and we assign to them the label *1*. Otherwise, they are labeled as *0*.

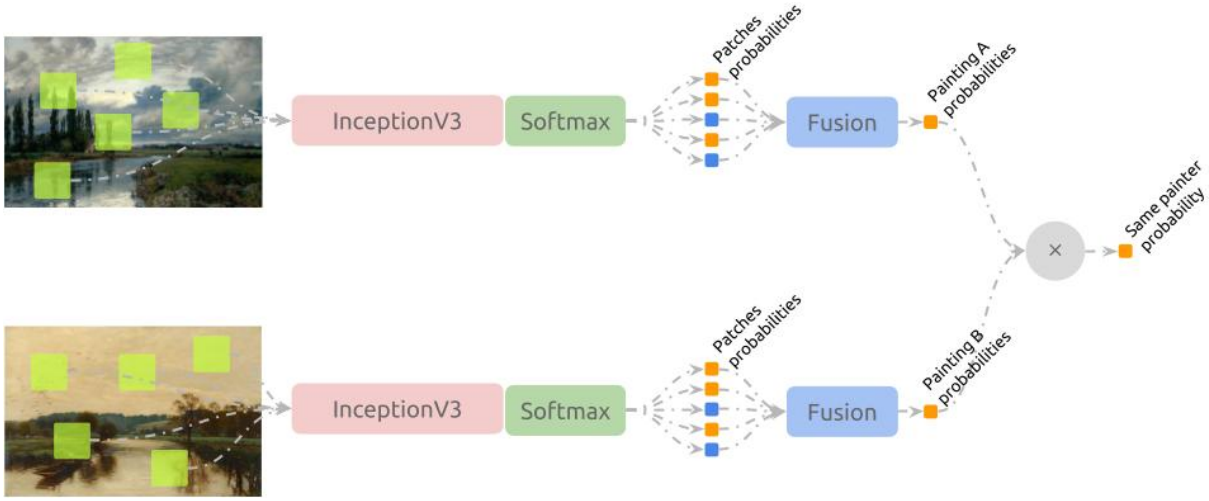


Figure 4.6: Diagram of the Siamese InceptionV3 network, jointed by the dot product operation.

This strategy strongly resembles the winning model at the competition, implemented by Nejc Ilenic[2]. However, a few key differences are visible. Firstly, our model does not resize images to match the conventional input sizes, as we consider this action potentially destructive on image quality, erasing much of the authors' signature patterns within the paintings. Additionally, we mitigate the performance drop down created by evaluating multiple patches by reusing most of the training from imagenet, only fine-tuning the last layer.

4.1.3.3 Siamese VGG16, Gram Matrix and Multiply Joint

Following ideas presented by Gatys et al. in [50], we constructed a classification model based on the style information extracted from painting patches, while also ignoring the content information of them.

Firstly, when building the L network, multiple layers of VGG16 — trained over imagenet — are selected to embed patches into multiple feature spaces. The style content is obtained as in [60], utilizing the Gram matrix of the output signal of each layer. The gram matrices are vectorized and concatenated, generating a single feature space which

[2]Leaderboard can be accessed at kaggle.com/c/painter-by-numbers/leaderboard

is fed to a *sigmoid* classifier. This architecture is listed in Algorithm 5 in the Appendix section. This pipeline is trained with binary cross-entropy, matching the painting patches with their respective one-hot encoded painter vector.

Two dropout and (untrained) dense layers are attached to L, guaranteeing embedding freedom to the system. The entire pipeline is then duplicated and jointed with the pairwise multiplication operation. A final *sigmoid* layer is used to discriminate matching and non-matching authorship patches. Once again, this architecture is detailed in Algorithm 6 in the Appendix section.

Figure 4.7 illustrates this model in its entirety.

softmax Network L ends with a *softmax* classifier instead of the *sigmoid* described above. L is trained with categorical cross-entropy instead.

200 Network L is trained to only distinguish between 200 random painters — the first 200 when sorted by their representing hash code —.

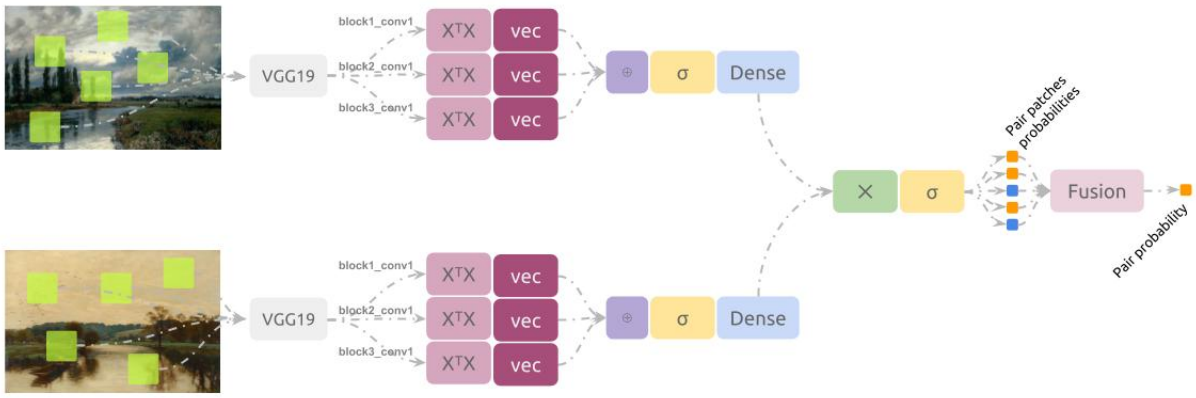


Figure 4.7: Diagram of the Siamese VGG16 network, jointed by the pairwise multiplication product operation.

4.1.3.4 Siamese InceptionV3, Embedding Dense Layers, l^2 Joint

This model re-utilizes the weights of the limbs trained in 4.1.3.2. However, it attaches two embedding layers containing 1024 units and *relu* activation to the output of the *softmax* function. This limb is then duplicated and jointed with the l^2 distance function. See Algorithm 7 in the Appendix for implementation details.

The entire model is trained using **contrastive loss**, which attempts to minimize the distances between samples of similar authorship while projecting the ones of significant difference far apart. This results in a network capable of projecting the data onto a space where the work of similar painters is concisely clustered.

Passing all 50 pairs of patches through the network results in 50 positive numbers usually between 0 and 1, which can be reduced to a single number t_i using the **mean** strategy. An answer if two paintings belong to a same painter can be finally reached by verifying if $min(max(1 - t_i, 0), 1) \geq 0.5$. Figure 4.8 summarizes this decision process.

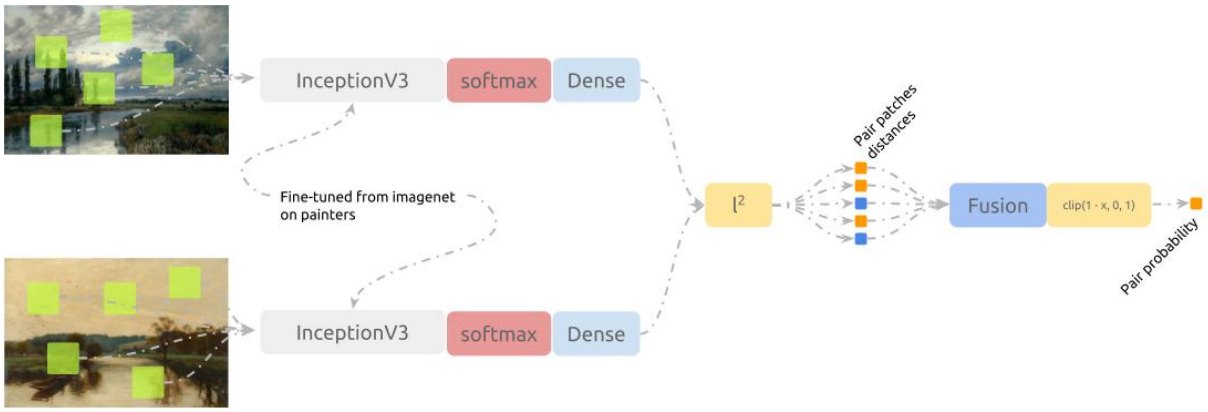


Figure 4.8: Diagram of the Siamese InceptionV3 network, jointed by the l^2 distance and trained with contrastive-loss.

4.1.3.5 Siamese InceptionV3, Embedding Dense Layers, Multiply Joint and Sigmoidal Activation

Again, trained weights of the feed-forward pipeline discussed in Subsection 4.1.3.1 are leveraged here. However, two embedding layers with *relu* activation are added at its end; the pipeline is then duplicated to receive two different patches and joined by a pairwise multiplication joint, which feeds a final unit with sigmoidal activation (Figure 4.9). Algorithm 8 in Appendix details this strategy. While inspired by the *dot* joint above, this setting eases the decision task of the system by replacing the "locked" correlation operation by a non-linear weighted mean reduction with trainable parameters. The network therefore has freedom to choose which painter recognizing outputs should be ignored when a pair of patches is presented.

The entire network is trained with binary cross entropy. While this will not spawn a comprehensible Euclidian embedding space, it can hopefully generate more accurate answers.

For each test case (a pair of paintings), the 50 pairs of extracted patches are randomly paired and fed to the network. Ensemble is performed using the **mean** strategy, and a matching prediction is made through the standard decision function $x \geq 0.5$.

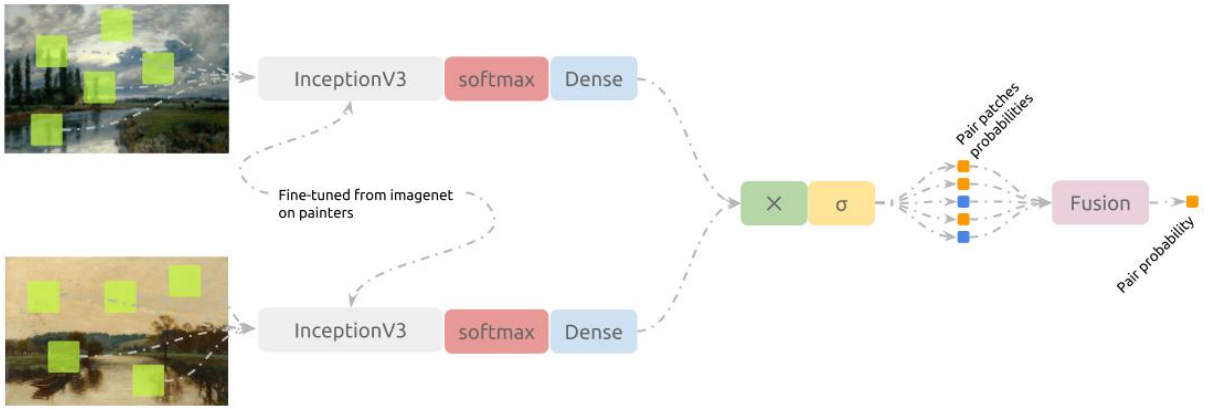


Figure 4.9: Diagram of the Siamese InceptionV3 network, jointed by the a pairwise multiplication.

4.1.3.6 Siamese Multilabel InceptionV3, Multiply Joint and Sigmoidal Activation

While authorship is the focused task in the competition, it is possible to inject more information in the training of the model, allowing it to make decisions over a broader perspective. As a first attempt, we propose to model the training of the limb network as a multi-label classification task.

As each painting $\boldsymbol{x}$ — and therefore each patch within that sample — in the training and validation sets is associated with an author, style and genre entry, it is possible to one-hot encode these labels as sparse vectors $\boldsymbol{p}_x \in \mathbb{R}^{1584}$, $\boldsymbol{s}_x \in \mathbb{R}^{135}$ and $\boldsymbol{g}_x \in \mathbb{R}^{42}$, respectively, and concatenate them into a single vector $\boldsymbol{y}_x \in \mathbb{R}^{1761}$, which will contain exactly three entries different than zero.

The limb network can be defined by prepending the convolutional layers of InceptionV3 to a dense layer with 1763 units and *sigmoidal* activation. This model can then be trained using binary cross-entropy, pairing each patch of x with its concatenated label vector y_x. When the limb network is trained, its output signal can be embedded and joined as in Subsection 4.1.3.5, and a decision can be reached for Painter-by-number test instances.

A simpler variation of this model (*dot*) is also attempted, where authorship matching is performed as in 4.1.3.2, considering only the output of network L (the concatenation of painter, style and genre information extracted from the patch). Results are available in Section 4.2.

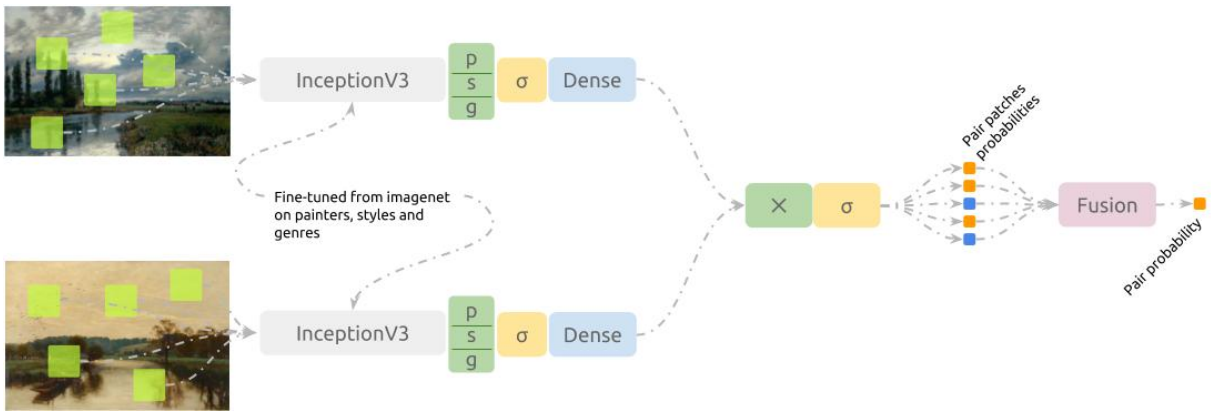


Figure 4.10: Diagram of the Multilabel Siamese InceptionV3 network.

4.1.3.7 Siamese InceptionV3, Multiple-Outputs, Multiple Joints and *Softmax* Activation

A second form of injecting more information into the decision process is to create a single-input, multiple-output network. In this layout, the model admits a single input patch. It propagates the signal generated by that patch through its sequential layers, which will eventually branch out in multiple parallel layers.

In this task, InceptionV3's convolutional pipeline is used as a base reduction network. Three dense layers with *softmax* activation were attached to its output (the result of the average pooling operation), designed to recognize the three groups of labels, painter, style and genre. During training, each output is paired with the actual one-hot encoded

labels and its categorical cross-entropy is computed. A final, differentiable loss function is then defined as the weighted sum of all partial losses, and minimized using *Adam* optimization [55]. Algorithm 9 in Appendix illustrates the limb network and its multiple outputs. Further details such layer-specific parameters as intermediate dropout layers were removed for brevity.

Once the limb is trained, its layers are "frozen" by masking the back-propagating gradient applied to these with 0, which will prevent any further updates to its weights. Dense embedding pipelines with two relu-activated layers are attached to each one of its outputs. The entire network is duplicated in order to receive a pair of patches and each pair of outputs is joined with the multiplication operation, followed by a single-unit sigmoidal activation (Algorithm 10 in Appendix).

Pairs of patches are fed to the network, following a balanced distribution of combinations among painters, and paired with three binary label groups corresponding to the matching information of the pairs (if these pairs belong to the same painter, style or genre).

A binary cross-entropy loss function is defined for each output, and the three are added to create the total loss function, which is minimized with *Adam* optimization [55] in a single training process. It is important to remark these outputs do not share their layers and do not affect each other's weights.

Once again, all existing layers are frozen. The three single-unit outputs of *mo* are concatenated and fed to one last dense layer with *sigmoid* activation (see Algorithm 11 for further details). This network is then trained with binary cross-entropy over a class-balanced distribution of patches' pairs, associated with its authorship matching information. When testing, patch ensemble and the test case decision is performed exactly as in 4.1.3.5 during test.

While this network is much larger — and slower — than the one described in Subsection 4.1.3.6, it offers an interesting feature: coherent semantic information can be extracted from each intermediate *softmax* and *sigmoid* units in the network. For instance, one might might use the limb *softmax* activations to classify a painting regarding its painter, style and genre, as well as to decide within two paintings share their creation year with $\text{linear}_{\text{year}}$; or use σ_2 and σ_3 to decide if two paintings belong to the same style and genre, respectively. Figure 4.11 summarizes the steps comprised in this strategy.

Additionally, we experiment with the following variations of this model. Their correspondent scores are also reported in Section 4.2.

sig The *sigmoid* activation function is used instead of the regular *softmax*. Binary cross-entropy is used to train each limb.

2048 e.u. Each layer in the artist embedding pipeline — previously containing 1024 units — is replaced by two 2048 unit layers.

IRNV2 The trained InceptionV3 base network is replaced by a trained InceptionResNetV2 model. Features are still extracted from the last global average pooling layer.

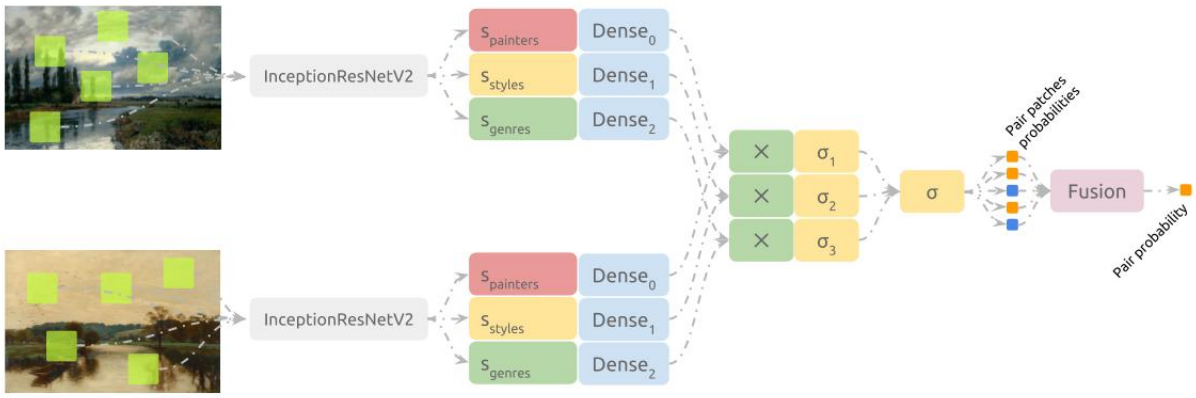


Figure 4.11: Diagram of the Siamese InceptionResNetV2 network.

4.2 Experiments and Results

Below we present results when applying the strategies enumerated in Section 4.1.3. Confusion matrix and AUC (the metric employed in the competition) are reported for each experiment, except by the first, which accuracy were not made available by the author.

The amount of patches extracted is indicated as a number f, followed by p.p. (per painting), indicating all paintings of each author were used and exactly f patches were extracted; or p.a. (per artist), entailing f patches were extracted from all possible paintings of each painter. The support in test is equal to the number of test cases (approximately 1 billion), for all methods.

model	**patches**	**tpr**	**tnr**	**AUC**
orange-nejc [3]	**1 p.a.**	**-**	**-**	**0.929**
M 4.1.3.1	50 p.p.	90%	25%	-
M 4.1.3.2	50 p.p.	100%	9%	0.903
M 4.1.3.3	1000 p.a.	63%	53%	0.614
M 4.1.3.4	50 p.p.	83%	64%	0.831
M 4.1.3.5	50 p.p.	76%	80%	0.87
M 4.1.3.6	**50 p.p.**	**93%**	**67%**	**0.914**
M 4.1.3.6, *dot*	50 p.p.	100%	7%	0.851
M 4.1.3.7	50 p.p.	92%	63%	0.898
M 4.1.3.7, sig	50 p.p.	88%	68%	0.884
M 4.1.3.7, 2048 e.u.	50 p.p.	100%	14%	0.874
M 4.1.3.7, IRNV2	50 p.p.	69%	78%	0.813

Table 4.2: Evaluation results for the models defined in Section 4.1 over Painter-by-Number test set.

[3] Nejc Ilenič (team id orange-nejc), first place in the competition's public leaderboard. His approach compresses the entire painting to fit the network input shape, generating a single input patch.

4.3 Discussions

In this section, we discuss the results observed in Section 4.2. Once again, the subsections are organized to match the model descriptions in subsection 4.1.3.

When available, a density distribution graph is presented to illustrate the activation distribution of the last layer of the model. Additionally, we provide the saliency activation map for some models, in order to verify the sections of the painting that most contributed to their predictions. All maps are generated using the guided backpropagation method as described by Springenberg et. al. [61], followed by the application of the absolute function. The output unit that is referred by the saliency map is always the most activating one, regardless if it describes the correct classification or not.

4.3.1 Strategies and Decision Models

Each following sections will discuss in details each model presented in subsection 4.1.3. Comparisons to the competition's winner — who achieved 0.929 AUC score with a network of custom architecture trained with paintings resized to match the size of the reception field 299×299 pixels — are drawn when pertinent.

4.3.1.1 InceptionV3, PCA, SVC and Equal Joint

While AUC is not available, the obtained 57.5% class-balanced accuracy indicates this model does not fare well on the problem. This is not surprising: compressing the paintings distances to the decision hyperplanes into single numbers and comparing them with the *Equal joint* discards important information regarding the distribution of the paintings in the embedded space created by the decision models. However, we observe that some small information pattern was indeed retained, as 25% of the positive samples are correctly classified.

4.3.1.2 Siamese InceptionV3, Dot Joint

The model presented a good AUC (0.0261 bellow the competition's winner), but a low class-balanced accuracy of 54.5% — and clearly overfitting onto the negative class —. This indicates the network rarely outputs above the canonical threshold 0.5 when fed with matching authorship pairs, but will likely trigger stronger signals in said scenario than when fed with non-matching authorship pairs. This pattern can be seen in the test output distribution plotted in Figure 4.12. Notwithstanding, this inconsistency can be fixed with a simple normalization, in which the output signal of the network would be scaled and shifted into $[0, 1]$.

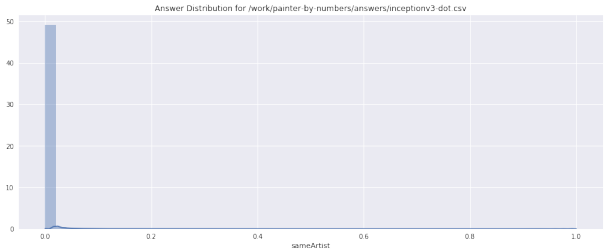
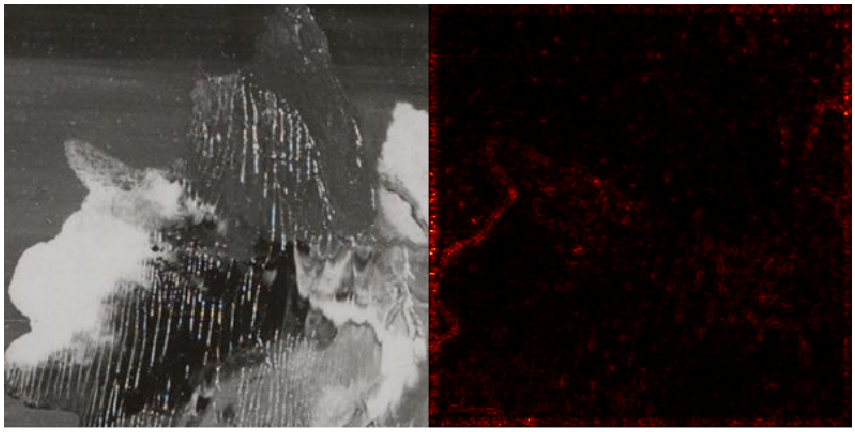
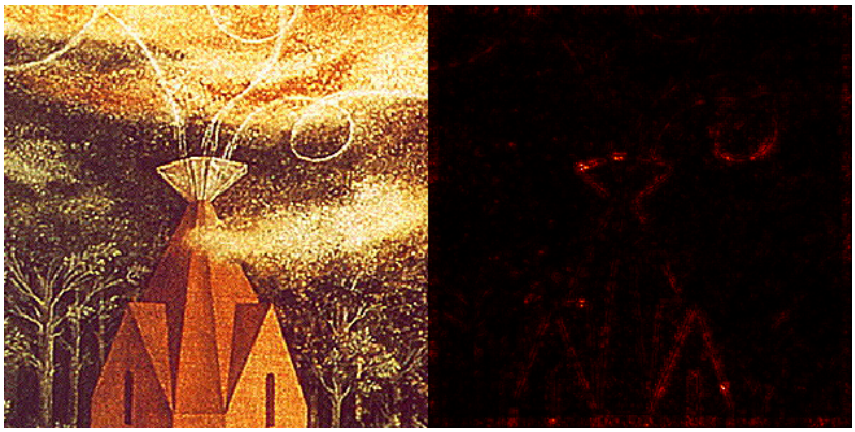


Figure 4.12: Activation density for the network, considering all samples (pairs) in the test set.

Figures 4.13 and 4.14 exemplifies patches (central crops) fed to the network and the saliency maps produced with respect to the label predicted by the network. These same maps were generated using the guided-backprop method. Regions associated with high absolute gradient value seem to intersect with detailed-rich and central regions of the patches: highly detailed faces; strong and abrupt variation in color or illumination; and sections of high contextual shift, where very divergent elements (with respect to appearance) meet.



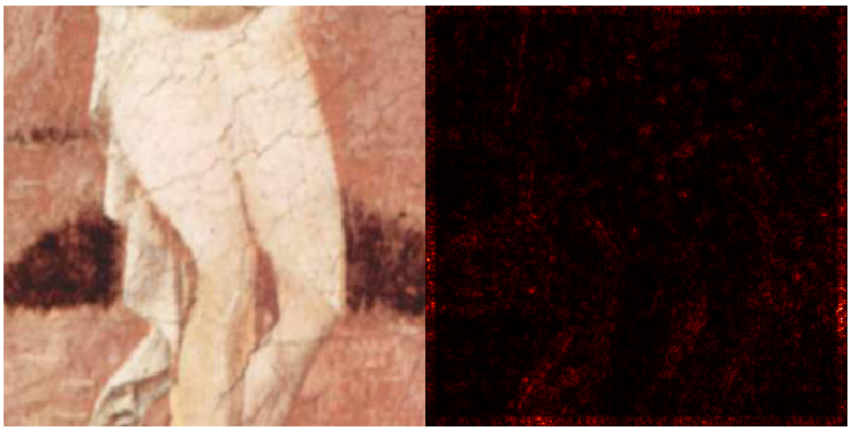
(a) “Untitled Vaporization" by Dolfi Trost, unknown year.



(b) “Microcosm" by Remedios Varo, 1959.

(c) “Catherine Mordvinova" by Carl-Ludwig Johann Christineck, 1773.

Figure 4.13: Examples of input painting patches and absolute gradient maps with respect to their maximum activation unit in the softmax classifying layer of the model L introduced in Subsection 4.1.3.2. All patches here were correctly classified by L. Therefore, the saliency maps highlight the regions in the input that most contributed to their correct prediction.



(a) “Christ on cross" by Paolo Uccello, 1438.

(b) “Still Life Bread and Leg of Lamb" by Paul Cezanne, 1866.

Figure 4.14: Examples of input painting patches and absolute gradient maps that were incorrectly classified by the L model presented in Subsection 4.1.3.2. Saliency maps highlight the regions in the input that most contributed to their incorrect prediction.

4.3.1.3 Siamese VGG16, Gram Matrix and Multiply Joint

This model presented a very low accuracy and AUC over the test set. The activation distribution shown in Figure 4.15 highlights the confusion of the model, which outputs most predictions close to the 50% decision threshold. A series of factors can be responsible for the poor performance of the model. Among them, we remark the limited visual field over VGG19 output signal, comprising only the first three layers, which might have prevented the model to assimilate higher-level features, essential for the classification process.

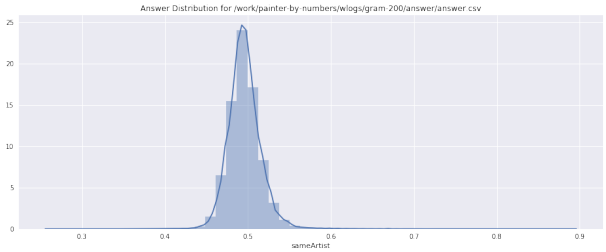


Figure 4.15: Activation density for the network, considering all samples (pairs) in the test set.

4.3.1.4 Siamese InceptionV3, Embedding Dense Layers, l^2 Joint

The model presented a fair class-balanced accuracy of 73.5% and 0.83 AUC. The output distribution spikes uncontrolled close to 0.0, which indicates many patches pairs are separated by a distance close (or greater) than 1, compared to the ones closely positioned. This might signify a region of great confusion for the model, in which a more detailed analysis might produce better results. For example, instead of simply considering two patches to have the same authorship if they differ by a distance smaller than 0.5, one could embed the training patches in the output space and extract authorship clusters. Finally, a test case could be performed by comparing the distances from the input patches to the clusters extracted.

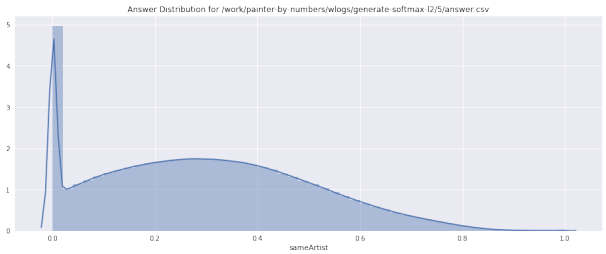


Figure 4.16: Activation density for the network, considering all samples (pairs) in the test set.

4.3.1.5 Siamese InceptionV3, Embedding Dense Layers, Multiply Joint and Sigmoidal Activation

Although this model did not perform well in the competition (due to its low AUC), it is the one with highest class-balanced classification score, due to the network being directly fit to answer for authorship matching. It also presents less concentrated output distribution, indicating many samples were classified close to the indecision threshold of 50%. Answers close to 0% appear more frequently than the ones close to 100% simply due to the large binary unbalance of the test set.

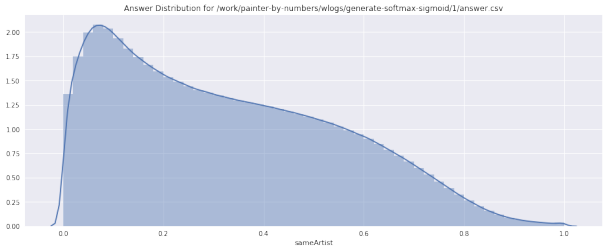


Figure 4.17: Activation density for the network, considering all samples (pairs) in the test set.

4.3.1.6 Siamese Multilabel InceptionV3, Multiply Joint and Sigmoidal Activation

This model presented the highest AUC score (0.914) among all designed in this research. As the main difference between this model and the previous is the inclusion of style and genre information when fitting the limb network, this result provides encouraging empirical evidence that model concerns with the authorship-matching problem can benefit themselves when mixing the multiple adjacent groups attached to paintings (artists, style and genre).

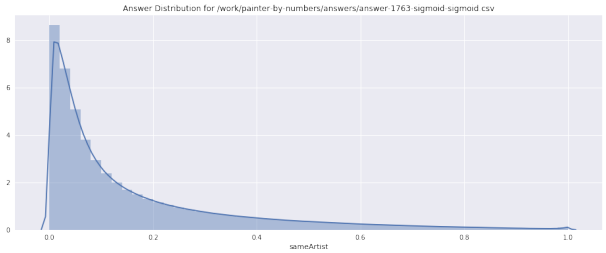


Figure 4.18: Activation density for the network, considering all samples (pairs) in the test set.

The dot variation, on the other hand, displayed a AUC score that was 0.063 below the base model, indicating the paramount necessity of pondering on the importance of the many different painters, styles and genres with the last sigmoid layer, instead of directly correlating the signals extracted from two patches.

4.3.1.7 Siamese InceptionV3, Multiple Outputs, Multiple Joints and Sigmoidal Activation

While achieving a good AUC score, 0.90, this setup was not able to outperform our best result shown in Section 4.1.3.6, nor the challenge winner. Furthermore, this model resulted in a true positive rate of 63% — 4% lower than the one shown in Subsection 4.1.3.6 —. This is likely due to the final matching unit in this model being fed by the combination of the artist, style and genre information in its reduced form (three sigmoid units), in opposite of being connected to a much richer feature space (one-hot encoding vectors). It is also worth remarking that, given the greater complexity of this model, a lower batch size of 48 patches (in opposite of the 128 batch size employed by model shown in

Subsection 4.1.3.2) was required to contain the training data in a single processing device. This limitation may have induced more noise in the training data, leading to a lower score.

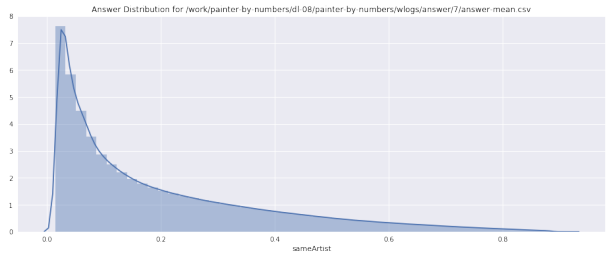
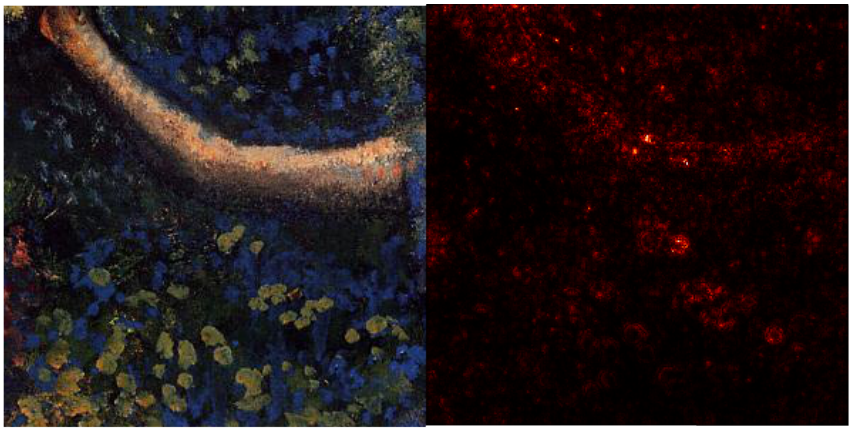


Figure 4.19: Activation density for the network, considering all samples (pairs) in the test set.

Once again, we observe (Figure 4.20) that high-contributing regions are incident on strong strokes, separating sections with very distinct light or colors. The patch presented in Figure 4.21a contains the exact opposite: a large section of homogeneous paint, without distinct strokes or color shifts.

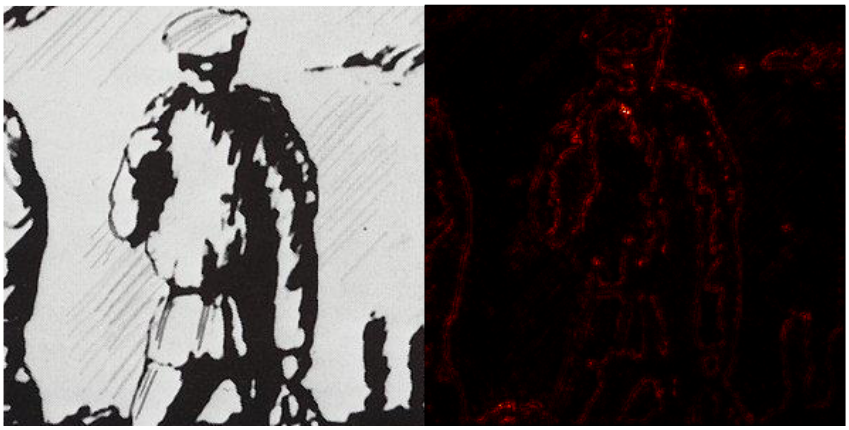
Martin Schongauer's painting, "Elephant in Hortus sanitatis", has been misclassified for Ilya Repin's work. We remark that Schongauer is extremely underrepresented in the dataset and his additional samples do not seem to share the artistic style from this one. On the other hand, Repin contains a massive amount of paintings, from which many are drawings and sketches. This leads us to conclude this mistake was caused by the strong unbalance in the data, entailing a stronger importance on discriminating Repin's samples rather than Schongauer's.

Although models described in Sections 4.1.3.7 *sig* and 4.1.3.7 *2048 e.u.* came close to its base score of 0.90, none of the variations of this model have achieved a higher AUC score. 4.1.3.7 IRNV2 presented the worst results: since from the start, when training L, the *2048 e.u.* variation presented a lower validation artist accuracy (25%) when compared to its base model 4.1.3.7 (46%), which indicates this variation possesses a too complex embedding system for the problem, difficult to train and easily subject to overfitting.



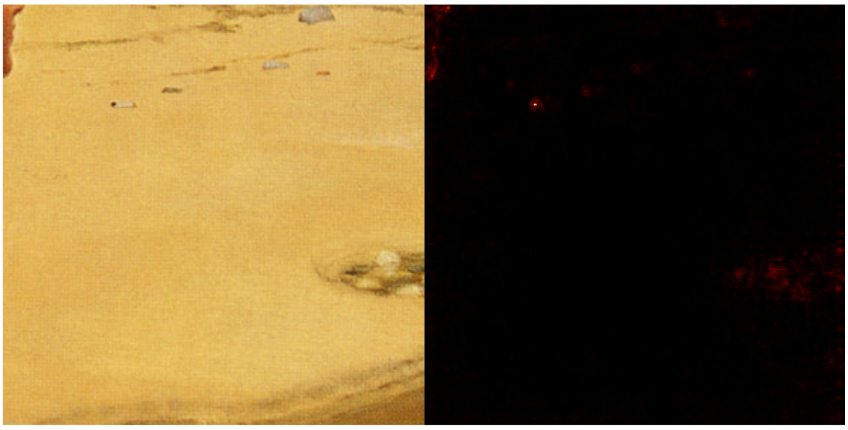
(a) “Dancer and Tambourine" by Edgar Degas, c.1897.

(b) “Among the Vines near Louveciennes" by Alfred Sisley, 1874.



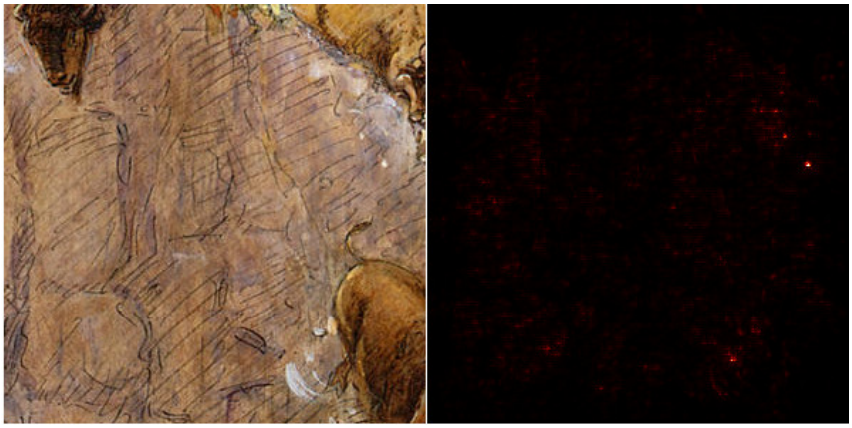
(c) “From Fishing" by Boris Kustodiev, 1923.

Figure 4.20: Examples of input painting patches and absolute gradient maps with respect to their maximum activation unit in the *artist* softmax classifying layer of the model L introduced in Subsection 4.1.3.7. All patches here were correctly classified by L. Therefore, the saliency maps highlight the regions in the input that most contributed to their correct prediction.



(a) “The Land Baby" by John Collier, 1899.

(b) “Elephant in Hortus sanitatis" by Martin Schongauer.



(c) “Driving Buffalo Over the Cliff" by Charles M. Russell, 1914.

Figure 4.21: Examples of input painting patches and absolute gradient maps that were incorrectly classified by the 4.1.3.7 L model. Saliency maps highlight the regions in the input that most contributed to their incorrect prediction.

Chapter 5

Conclusions

With the great infusion of art in our daily lives, flowing through informal streams from inconsistent sources and capturing conditions, art samples constitute vast and heterogeneous data sets. This, already with the high number of styles and art schools, forces researching groups interested in the authorship-matching and recognition problems to not only push for models that can outperform the state of the art with greater test accuracy, but to consider multiple adjacent goals as well. As examples, we mention the decrease in data requirement and computational performance; the reproducibility of accuracy in cross-dataset experiments; and the explainability of the model produced.

In this work, we have experimented with multiple data-driven approaches that formed viable solutions for the authorship-matching and recognition problems, considering aspects such as domain knowledge, data requirement, computational performance and cross-dataset reproducibility.

In the scope of authorship attribution for van Gogh paintings, we composed the ideas of limited patch sampling, transfer learning, SVM and fusion (by ensembling the classification results) to create a machine-learning model capable of outperforming the literature's results, while drastically reducing memory requirements and capable of fair generalization onto cross-domain datasets. More specifically, we achieved a 96.5% class-balanced accuracy score in the original van Gogh data test set while reducing training time from three hours to only 18 minutes.

We observed the model did not perform well over a small set composed of scanned, high-definition recaptures of the test paintings, and we concluded that although the set was composed by difficult samples, features extracted from the scanned samples clearly distinguished from photographed ones. On the other hand, when tested over recaptures extracted from multiple sources and unseen samples extracted from the van Gogh museum website, the model scored a class-balanced accuracy of 82% and 72%, respectively, effectively demonstrating its generalization capacity.

We have found that fusing the classification results of multiple recaptures of a single painting (after combining their respective patches' results) further strengthened the average test accuracy of a decision model to 87% (five points increase), suggesting data-driven art analysis tools should not disregard the currently heterogeneous state-of-art data, but instead leverage these differences in order to build stability.

When considering provenance for multiple painters, we developed decision models

capable of competitive AUC scores, when compared to the ones reported in the public leaderboard on Kaggle. We remark that in all strategies employed, paintings fed to the models were not scaled as performed by other competitors, as we deemed this procedure to erase much of the authorship information in the painting and highlight artificial differences among them. Among the developed models, we highlight the three below:

1. The model based on painter-classification and dot product of feature-vectors representing the embedded patches, which is described in Subsection 4.1.3.2. This model reached 0.903 AUC and 57.5% class-balanced accuracy, indicating strong discriminating capabilities but a shifted class-decision threshold. Closer inspection of saliency maps on a few samples have demonstrated abrupt changes in color or illumination represented decisive sections when determining authorship.

2. The strategy that concatenated painter, style and genre information into a single feature-vector and fed it to a non-linear decision network (Subsection 4.1.3.6). This model presented the best AUC (0.914) — being only 0.01528 points bellow the first placed — and class-balanced accuracy (80%). This is unsurprising, as the combined information enriched the decision capability of the model, while maintaining a relative simplicity that entailed a high batch size and a greater stability in the training process.

3. The model shown in Subsection 4.1.3.7, in which a more complex parallel architecture is adopted to combine painter, style and genre information in a more organized manner. This strategy has shown a promising AUC of 0.898 and class-balanced accuracy of 77.5%, but failed to outperform the model presented in Subsection 4.1.3.6, as information was combined in a more digested form (three decision units) instead of a richer combination of feature-spaces and batch size was severally impacted by the complexity attached to this network. We inspected once more that high-contributing regions coincide with strong strokes and varying sections of the paintings.

On the technical side, we have become acquainted with the deep learning Keras project, which was the main library employed during the tasks described in this work. As challenges were presented during this work, further improvements to the tool became visible. This translated into a brief set of contributions made to the original repository[1] and answers in stackoverflow[2].

Finally, we foresee different lines of improvement applicable to this work:

1. It would be possible to increase the available data and mitigate the class-unbalance problem of art data sets by synthesizing patches using generative models, such as GANs [62], CANs [63] or auto-encoders [28].

2. Methods of attention could be employed to focus on relevant areas of a painting, in opposite of simple random patch extractions. For example, after an initial, conventional feed-forward network is trained over a dataset with many painters, saliency

[1] https://github.com/keras-team/keras/issues?q=involves:lucasdavid
[2] https://stackoverflow.com/users/2429640/ldavid?tab=answers

maps relative to the maximization of multiple random output units could be extracted over an input painting. A probability distribution vector could then be build by averaging the absolute values of these (vectorized) maps, and patches extracted following such distribution. Repeating this procedure to all paintings would produce a new dataset, containing samples that represented the most probably activating regions (of most interest) of the input paintings. A second model could then be trained and tested over this cleaner set.

3. As the datasets used here mainly comprised acclaimed artists, a more extensive and strict set of tests could be applied. To better understand the behavior of our models, it is paramount to also consider more contemporary and less frequently occurring art styles and painters, as well as multiple forgeries and replicating styles for both known and unknown artists. Only then, we will be able to measure their capacity of distinguishing authorship in more realistic (fuzzier) scenarios, further fortifying it.

Bibliography

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] E. Panofsky, *Meaning in the Visual Arts*, ser. Art History. University of Chicago Press, 1955.

[3] B. Grosvenor. (2012) On connoisseurship. [Online]. Available: http://arthistorynews.com/articles/1101_On_connoisseurship

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. Adaptive Computation and Machine Learning series. The MIT Press, 2016.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[7] U. Stańczyk and K. A. Cyran, "Machine learning approach to authorship attribution of literary texts," *International Journal of Applied Mathematics and Informatics*, vol. 1, no. 4, pp. 151–158, 2007.

[8] N. Khanna, A. K. Mikkilineni, and E. J. Delp, "Scanner identification using feature-based processing and analysis," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 1, pp. 123–139, 2009.

[9] A. Ferreira, L. C. Navarro, G. Pinheiro, J. A. dos Santos, and A. Rocha, "Laser printer attribution: Exploring new features and beyond," *Forensic science international*, vol. 247, pp. 105–125, 2015.

[10] M. Campbell, A. J. Hoane, and F.-h. Hsu, "Deep blue," *Artificial intelligence*, vol. 134, no. 1, pp. 57–83, 2002.

[11] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[13] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[14] C. Pike-Burke, "Multi-objective optimization," 2019. [Online]. Available: https://www.lancaster.ac.uk/pg/pikeburc/report1.pdf

[15] S. Lyu, D. Rockmore, and H. Farid, "A digital technique for art authentication," *National Academy of Sciences of the United States of America*, vol. 101, no. 49, pp. 17 006–17 010, 2004.

[16] M. Bressan, C. Cifarelli, and F. Perronnin, "An analysis of the relationship between painters based on their work," in *15th IEEE International Conference on Image Processing*. IEEE, 2008, pp. 113–116.

[17] H. Qi, A. Taeb, and S. M. Hughes, "Visual stylometry using background selection and wavelet-hmt-based fisher information distances for attribution and dating of impressionist paintings," *Signal Processing*, vol. 93, no. 3, pp. 541–553, 2013.

[18] S. Agrawal and O. Sahu, "Two-channel quadrature mirror filter bank: An overview," *ISRN Signal Processing*, vol. 2013, 2013.

[19] T. Cox and M. Cox, *Multidimensional Scaling, Second Edition*, ser. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. CRC Press, 2000.

[20] D. G. Lowe *et al.*, "Object recognition from local scale-invariant features." in *iccv*, vol. 99, no. 2, 1999, pp. 1150–1157.

[21] S. I. Costa, S. A. Santos, and J. E. Strapasson, "Fisher information distance: a geometrical reading," *Discrete Applied Mathematics*, vol. 197, pp. 59–69, 2015.

[22] Wikiart.org: Visual art encyclopedia. [Online]. Available: http://www.wikiart.org/

[23] C. R. Johnson, E. Hendriks, I. J. Berezhnoy, E. Brevdo, S. M. Hughes, I. Daubechies, J. Li, E. Postma, and J. Z. Wang, "Image processing for artist identification," *IEEE Signal Processing Magazine*, vol. 25, no. 4, pp. 37–48, 2008.

[24] P. Blunsom, "Hidden markov models," 2004, Utah State University. [Online]. Available: http://digital.cs.usu.edu/~cyan/CS7960/hmm-tutorial.pdf

[25] M. Irfan and D. G. Stork, "Multiple visual features for the computer authentication of jackson pollock's drip paintings: beyond box counting and fractals," in *IS&T/SPIE Electronic Imaging*, 2009, pp. 72 510Q–72 510Q.

[26] H. Qi and S. Hughes, "A new method for visual stylometry on impressionist paintings," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2011, pp. 2036–2039.

[27] H. Liu, R. H. Chan, and Y. Yao, "Geometric tight frame based stylometry for art authentication of van gogh paintings," *Applied and Computational Harmonic Analysis*, 2015.

[28] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks*. Springer, 2011, pp. 52–59.

[29] C. Montagner, R. Jesus, N. Correia, M. Vilarigues, R. Macedo, and M. J. Melo, "Features combination for art authentication studies: brushstroke and materials analysis of Amadeo de Souza-Cardoso," *Multimedia Tools and Applications*, vol. 75, no. 7, pp. 4039–4063, 2016.

[30] Z. Liu, J. Cai, and R. Buse, *Handwriting Recognition: Soft Computing and Probabilistic Approaches*, ser. Studies in Fuzziness and Soft Computing. Springer Berlin Heidelberg, 2003.

[31] G. Folego, O. Gomes, and A. Rocha, "From impressionism to expressionism: Automatically identifying van gogh's paintings," in *IEEE International Conference on Image Processing*. IEEE, 2016, pp. 141–145.

[32] L. Shamir, T. Macura, N. Orlov, D. M. Eckley, and I. G. Goldberg, "Impressionism, expressionism, surrealism: Automated recognition of painters and schools of art," *ACM Transactions on Applied Perception*, vol. 7, no. 2, p. 8, 2010.

[33] C. Liu and H. Jiang, "Classification of traditional chinese paintings based on supervised learning methods," in *IEEE International Conference on Signal Processing, Communications and Computing*. IEEE, 2014, pp. 641–644.

[34] E. Cetinic and S. Grgic, "Automated painter recognition based on image feature extraction," in *55th International Symposium ELMAR*. IEEE, 2013, pp. 19–22.

[35] M. Sun, D. Zhang, J. Ren, Z. Wang, and J. S. Jin, "Brushstroke based sparse hybrid convolutional neural networks for author classification of chinese ink-wash paintings," in *IEEE International Conference on Image Processing*. IEEE, 2015, pp. 626–630.

[36] C. Thomas and A. Kovashka, "Who's behind the camera? identifying the authorship of a photograph," *arXiv preprint arXiv:1508.05038*, 2015.

[37] N. van Noord, E. Hendriks, and E. Postma, "Toward discovery of the artist's style: Learning to recognize artists by their artworks," *IEEE Signal Processing Magazine*, vol. 32, no. 4, pp. 46–54, 2015.

[38] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in neural information processing systems*, 2014, pp. 487–495.

[39] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.

[40] T. Mensink and J. Van Gemert, "The rijksmuseum challenge: Museum-centered visual recognition," in *International Conference on Multimedia Retrieval*. ACM, 2014, p. 451.

[41] N. Viswanathan, "Artist identification with convolutional neural networks," vol. 77, pp. 89–8, 2017.

[42] J. Chen and A. Deng, "Comparison of machine learning techniques for artist identification," 2018.

[43] J. Zujovic, L. Gandy, S. Friedman, B. Pardo, and T. N. Pappas, "Classifying paintings by artistic genre: An analysis of features & classifiers," in *IEEE International Workshop on Multimedia Signal Processing*. IEEE, 2009, pp. 1–5.

[44] A. Kovashka and M. Lease, "Human and machine detection of stylistic similarity in art," *CrowdConf*, vol. 2010, 2010.

[45] P. Bajcsy and M. Moslemi, "Discovering salient characteristics of authors of artworks," in *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2010, pp. 75 310B–75 310B.

[46] R. S. Arora, "Towards automated classification of fine-art painting style: A comparative study," Ph.D. dissertation, Rutgers University-Graduate School-New Brunswick, 2012.

[47] Y. Bar, N. Levy, and L. Wolf, "Classification of artistic styles using binarized features derived from a deep neural network," in *Workshop at the European Conference on Computer Vision*. Springer, 2014, pp. 71–84.

[48] B. Saleh and A. Elgammal, "A unified framework for painting classification," in *IEEE International Conference on Data Mining Workshop*. IEEE, 2015, pp. 1254–1261.

[49] A. Bergamo, L. Torresani, and A. W. Fitzgibbon, "Picodes: Learning a compact code for novel-category recognition," in *Advances in Neural Information Processing Systems*, 2011, pp. 2088–2096.

[50] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[51] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint*, 2017.

[52] Z. Wang, L. Zhao, W. Xing, and D. Lu, "Glstylenet: Higher quality style transfer combining global and local pyramid features," *CoRR*, vol. abs/1811.07260, 2018.

[53] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[54] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[56] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[57] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[58] (2006) Van gogh museum - overzicht alles over vincent. [Online]. Available: https://vangoghmuseum.nl

[59] Painter by numbers: Does every painter leave a fingerprint? Kaggle. [Online]. Available: https://www.kaggle.com/c/painter-by-numbers

[60] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks," *arXiv preprint arXiv:1505.07376*, vol. 12, 2015.

[61] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[62] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[63] A. Elgammal, B. Liu, M. Elhoseiny, and M. Mazzone, "Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms," *arXiv preprint arXiv:1706.07068*, 2017.

Appendix A

Algorithms Used for Authorship Attribution in van Gogh Paintings

```
def extractor(dataset_files):
  x = []
  for sample in dataset_files:
    painting = img_to_array(load_img(sample).convert('HSV'))
    patches = extract_patches(painting, 50, 'random')
    histograms = [np.histogram(x[:, :, 2], bins=64, range=(0, 255))
                    .flatten()
                  for x in patches]
    x = np.concatenate(x, histograms)
  return x
```

Listing 1: Histogramming of colors from painting image files.

```
x = Input(shape=[299, 299, 3])

base = InceptionV3(x, weights='imagenet')
y = base.get_layer('global_average_pooling2d_1').output
y = Dense(2, activation='softmax')(y)

model = Model(inputs=x, outputs=y)
```

Listing 2: Painting classifier that uses a pre-trained InceptionV3 and a dense *softmax* classifier.

```
x = Input(shape=[299, 299, 3])

base = InceptionV3(x, weights='imagenet')
y = base.get_layer('global_average_pooling2d_1').output
extractor = Model(inputs=x, outputs=y)

model = Pipeline([
  ('reducer', PCA()),
  ('clf', SVC())
])

grid = GridSearchCV(model, params, cv=3)
```

Listing 3: Painting classifier based on pre-trained InceptionV3 attached to a support vector machine classifier.

```
def l2(inputs):
    ya, yb = inputs
    return K.sqrt(K.maximum(
        K.sum(K.square(ya - yb), axis=1, keepdims=True),
        K.epsilon()))

xa, xb = Input(shape=[299, 299, 3]), Input(shape[299, 299, 3])

base = VGG19(xa, weights='imagenet')
ya = base.get_layer('flatten').output
L = Model(inputs=xa, outputs=ya)

yb = L(xb) # duplicates entire network, reusing all layers
y = Lambda(l2)([ya, yb])

model = Model(inputs=[xa, xb], outputs=y)
```

Listing 4: Pair-painting discriminator based on VGG19 and contrastive dissimilarity.

Appendix B

Algorithms Used for Provenance Analysis for Multiple Painters

```
def gram(x):
    x = K.permute_dimensions(x, (0, 3, 1, 2))
    B, C, H, W = K.shape(x)
    features = K.reshape(x, K.stack([B, C, H * W]))
    return K.batch_dot(features, features, axes=2)

style_layers = ('block1_conv1', 'block2_conv1', 'block3_conv1')
ys = []

x = Input(shape=[299, 299, 3])

base = VGG16(x, weights='imagenet')

for l in style_layers:
  y = base.get_layer(l).output
  y = Lambda(gram)(y)
  y = Flatten()(y)
  ys.append(y)

y = concatenate(ys)
y = Dense(1584, activation='sigmoid')(y)

L = Model(inputs=x, outputs=y)
```

Listing 5: Painting classified based on the gram-matrix of VGG19 activations when presented with paintings.

```
y = L.output
y = Dropout(.4)(y)
y = Dense(2028, activation='relu')(y)
y = Dropout(.4)(y)
y = Dense(2028, activation='relu')(y)
L = Model(inputs=L.inputs, outputs=y)

xa, xb = Input(shape=[299, 299, 3]), Input(shape=[299, 299, 3])
ya, yb = L(xa), L(xb)

y = multiply([ya, yb])
y = Dense(1, activation='sigmoid')(y)
model = Model(inputs=[xa, xb], outputs=y)
```

Listing 6: Pair-painting discriminator based on the classifier described in Algorithm 5.

```
def l2(inputs):
    ya, yb = inputs
    return K.sqrt(K.maximum(
        K.sum(K.square(ya - yb), axis=1, keepdims=True),
        K.epsilon()))

xa, xb = Input(shape=[299, 299, 3]), Input(shape[299, 299, 3])

base = InceptionV3(xa, weights='imagenet')
y = base.get_layer('global_average_pooling2d_1').output
y = Dense(1024, activation='relu')(y)
ya = Dense(1024, activation='relu')(y)
L = Model(inputs=xa, outputs=ya)

yb = L(xb)  # duplicates entire network, reusing all layers
y = Lambda(l2)([ya, yb])

model = Model(inputs=[xa, xb], outputs=y)
```

Listing 7: Pair-painting discriminator based on l^2-contrastive differences in feature space generated by InceptionV3 pre-trained on Imagenet [53].

```
xa, xb = Input(shape=[299, 299, 3]), Input(shape[299, 299, 3])

base = InceptionV3(xa, weights='imagenet')
y = base.get_layer('global_average_pooling2d_1').output
y = Dense(1024, activation='relu')(y)
ya = Dense(1024, activation='relu')(y)
L = Model(inputs=x, outputs=ya)
yb = L(xb)  # duplicates entire network, reusing all layers
y = multiply([ya, yb])
y = Dense(1, activation='sigmoid')(y)

model = Model(inputs=[xa, xb], outputs=y)
```

Listing 8: Pair-painting discriminator based on InceptionV3, multiply joint and and sigmoidal activation.

```
x = Input(shape=[299, 299, 3])
base = InceptionResNetV2(x, weights='imagenet')
y = base.get_layer('global_average_pooling2d_1').output
p = Dense(1584, activation='softmax', name='painter')(y)
s = Dense(156, activation='softmax', name='style')(y)
g = Dense(42, activation='softmax', name='genre')(y)
limb = Model(inputs=x, outputs=[p, s, g])
```

Listing 9: Multiple-outputs classifier based on InceptionV3 pre-trained on Imagenet [53].

```
outputs = []
for y, units in ((p, 1024), (s, 256), (g, 256)):
  y = Dense(units, activation='relu')(y)
  y = Dense(units, activation='sigmoid')(y)
  outputs.append(y)

a, b = Input(shape=[299, 299, 3]), Input(shape=[299, 299, 3])
extended_limb = Model(inputs=x, outputs=outputs)
la, lb = extended_limb(a), extended_limb(b)

outputs = []
for layer, ya, yb in zip(('painter', 'style', 'genre'), la, lb):
  y = Multiply()(ya, yb)
  y = Dense(1, activation='sigmoid')(y)
  outputs.append(y)

mo = Model(inputs=[a, b], outputs=outputs)
```

Listing 10: Pair-painting discriminators with respect to painter, style and genre.

```
y = Concatenate(axis=1)(mo.outputs)
y = Dense(1, activation='sigmoid')(y)
model = Model(inputs=[a, b], outputs=y)
```

Listing 11: Pair-painting that combines author, style and genre information matching to discriminate authorship.